End-user Programming of Ambient Narratives for Smart Retail Environments

ISBN 978-90-386-1527-1

“Men’s Native Title Painting, 1998”
Reproduced from the cover of Pila Nguru - the Spinifex People
by Scott Cane (Fremantle Arts Centre Press, 2002).
Reproduced by arrangement with the publisher.

Cover design by Paul Verspaget & Carin Bruinink

The work described in this thesis has been carried out at the Philips Research Laboratories Eindhoven, The Netherlands, as part of the Philips Research Programme.

End-user Programming of Ambient Narratives for Smart Retail Environments

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van
de Rector Magnificus, prof.dr.ir. C.J. van Duijn,
voor een commissie aangewezen door het College
voor Promoties in het openbaar te verdedigen op
dinsdag 17 maart 2009 om 16.00

door

Markus Gerardus Leonardus Maria van Doorn

geboren te Veghel

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. E.H.L. Aarts
en
prof.dr.ir. A.P. de Vries

Contents

Preface and Acknowledgements

When you put a thing in order, and give it a name, and you are all in accord, it becomes – From the Navaho, Masked Gods, Waters, 1950

The start of this research goes back many years. In 1999 after I graduated from the University of Twente on the use of visual thesauri in multimedia information retrieval, I was drawn to Philips Research that was forming its ideas on ambient intelligence, a vision that puts the user experience in the foreground and technology in the background. Philips Research and the New Media Systems and Applications group turned out to be an exciting place to work in. The culture was open and informal, the projects highly innovative and the teams multi-disciplinary. In 2000 I moved away from multimedia information retrieval to presentation generation. The Internet had become serious business and Philips was quick to see that all kinds of consumer electronic devices with different capabilities would connect to this Internet. Since these different devices would have different presentation capabilities, we would need to adapt the presentation of the personalized music, media, content to the user and his/her device. Not much later context entered into the picture and we looked at how Semantic Web technology could be used to determine which music should be presented in the environment for which user.

In that same period the user-centric design philosophy became increasingly popular in Philips. In 2002 HomeLab, a facility to test and evaluate ambient intelligent concepts with real users in a natural surroundings opened. HomeLab and the user-centric design approach was a success but also had its downsides. We spent many hours designing and building systems that were suitable to do one or a few things well, but if you would want something different, you needed to start over and build a new demo. In the early days of ambient intelligence the excitement and novelty of these early demonstrations clouded this aspect from sight, but one day the excitement would diminish and people would be asking us to deliver intelligent environments that could support a wide-range of ambient applications at relatively low-cost.

The user-centered design philosophy would not lead to an answer of this question. We needed an integral, holistic design philosophy that would look at experience from many different angles including philosophy, cognitive psychology, sociology, economics, performance and literary studies. What is experience? What does it mean to experience something? What are the reasons why experience plays such an important role in today's society and economy? What is the role of play and performance in culture? How can ambient intelligence support these everyday life performances? What is the relation between performance, text and interactivity in everyday life?

There were times when I thought this research would never condense into a coherent whole, but if you hold on long enough to a specific problem and persist, the solution will eventually form inside your mind and explain itself to you. This book that lies in front of you is the result of that process.

This work would never have been possible without the help of many people. The first word of thanks goes to my supervisors Emile Aarts and Arjen de Vries. Arjen not only supervised this PhD research but also my master thesis in 1999. He was the first to make me consider the idea of pursuing a PhD back at the University of Twente and convinced me to do so. Without his motivation, support and patience this work would not have been possible. Thank you Emile for all these many years you gave me excellent guidance and helped me to stay focussed. I am honoured that Prof. Anton Nijholt, Dr. Lynda Hardman and Prof. Matthias Rauterberg agreed to kindly join my committee.

Many (ex-)colleagues of the New Media Systems and Applications department have helped me throughout the years in the research that led up to this thesis. Ramon Clout, Richard Doornbos, Warner ten Kate, Herman ter Horst, Natasha Kravtsova played an imporant role in shaping the earlier work. Special thanks go to Evert van Loenen as projectleader of the Dreamscreen project, of which the intelligent shop window is part, and Vic Teeven, who as manager of the ExperienceLab provided a stable home for the intelligent shop window installation.

Without the help of the ExperienceLab team, Lars Kaijser, Rick van Haasen, Han Kok, Tom Geelen, Roxana Frunza, Henk Sandkuyl, Arjan Claasen and Peter Jakobs of the Software Engineering Services department and Henk van der Weij of BigCamp Multimedia the ambient narrative shop window system and its authoring environments would be an idea and not a working system. Thanks for coding and maintaining the software for all these years. I enjoyed the many meetings and discussions we had. Special thanks also to Herjan van den Heuvel for his work on context modelling for the ambient narrative engine, Tatiana Lashina for giving helpful comments and improvements on the editor user interface, Angelique Kessels and Jettie Hoonhout for helping me in setting up the user study in ShopLab.

I would also like to express my gratitude to Albert Boswijk, director of

the European Center for the Experience Economy, for sharing his knowledge on the characteristics and trends of the experience economy and playing an active role in finding retailers, designers and consultants for the user research. Thanks also to Anna Snel of the University of Amsterdam for pointing me to more great books and ideas.

Thanks to Philips Research management and in particular to Emile Aarts, Maurice Groten, Reinder Haakma for supporting the research that led to this thesis. I'd also like to thank all my colleagues at Philips Research who create such a stimulating working environment.

Most of all, I'd like to express my gratitude to Linda and my parents for all their love and support during the long time it took me to finish this project. It is to them that I would like to dedicate this work.

Mark van Doorn, 's-Hertogenbosch 2008

1

Introduction

The most fundamental technologies in our life are those that have disappeared into the background of our everyday life, so much integrated in our way of living that we simply take them for granted and forget how life was without them. Take writing for example: We find writing not only in books, papers, magazines, but also on street signs, maps, advertisements, packaged products and foods and so many other products and objects. We use this technology effortlessly and have become unaware of its presence. In a Scientific American article entitled *The Computer of the 21st century* [Weiser, 1991], Mark Weiser envisioned that the combination of ongoing miniaturization in the semiconductor industry and device interconnectivity would not only lead to better, faster computers but also to a new way of interacting with computers that could make computing as effortless and ubiquitous as writing: Instead of interacting with one computer at a time, Weiser envisioned a world where people would interact with many computers, in a wide variety different shapes and sizes, each suited for a particular task, at the same time. Rather than each fighting for our attention, these computing devices would move from the periphery to the center of our attention and back depending on our context.

Ambient intelligence [Aarts, Harwig & Schuurmans, 2002] refers to electronic environments that are sensitive and reactive to the presence of people and builds on this ubiquitous computing vision. The goal of ambient intelligent environments is to support people in their everyday life activities through technology and media. By positioning users and their needs in the center of attention and technology in a supporting role, ambient intelligence essentially adopts a user-centric design philosophy [Draper, Norman & Lewis, 1986] on ubiquitous computing.

Over the past ten years, many research prototypes and demonstrations of ambient intelligent systems have been developed, see e.g. [Dertouzos, 1999; Kindberg, Barton & Morgan, 2002; Aarts & Marzano, 2003]. Many of these examples are restricted in terms of application scope and usage. This has a number of reasons. Technologically, it has proven difficult to design a device or system in advance that is capable of dealing with the typically complex and dynamically changing environments that we encounter in everyday life. Multiple actors can be involved in several activities at once while using a variety of devices at the same time. Narrowing down the application domain is one way of dealing with this issue, but this complexity reduction can miss its target if it results in putting aside these characteristics of social environments. Moreover, a product-oriented, user-centered design approach has a natural tendency to discard all elements that do not contribute directly to the end-user experience or match current user needs. If these aspects are not taken into account as (future) user needs this can result in applications that are hard to reuse or difficult to extend over time [Abras, Maloney-Krichmar & Preece, 2004].

Rather than designing for the end-user and trying to address this complexity and uncertainty in advance, an alternative approach is to design the environment in such a way that it can easily be changed and programmed by end-users. In this way the initiative shifts from the designer to the user, which brings a number of advantages. First, it gives people the freedom to design the environment around them the way they see fit. If people want a professional to design this environment for them, they have at least the possibility to choose. This view also acknowledges the larger, broader social-economical and cultural trend of empowerment, of making people active participants in economical, political, and cultural decision making processes. Second, people are likely to put more trust in the smart environment if this environment works in a transparent, understandable and controllable way. Third, by establishing a continuous dialogue with the end-user, there is less of a burden on the system to determine the next (sequence of) actions autonomously. If the system encounters a conflict or is unsure about what to do next, it can simply ask the user for feedback and correct itself.

1.1 End-user Programming

Since the 1960s researchers have worked on a number of programming languages and environments with the goal to make programming more accessible to a larger group of people. These end-user programming environments typically fall into two large groups [Kelleher & Pausch, 2005]: Systems that teach people how to program (teaching systems) and systems that empower people to build things that are tailored towards their needs (empowering systems), which are of particular interest to us. Well-known examples of systems of this latter category are spreadsheets, statistical packages, CAD/CAM systems and web authoring tools. It has turned out that users of these systems are willing to invest time in learning programming languages and environments if that helps them to get their tasks done in a more effective or efficient way [Nardi, 1993]. There are many ways of making programming more accessible to people. Research on end-user programming has therefore looked at a wide variety of topics, including human computer interaction, computer graphics and visualization, programming languages, programming models and strategies, user modelling, empirical studies, computer supported collaborative work, and more recently software engineering [Burnett, Engels, Myers & Rothermel, 2007]. In terms of application areas, ubiquitous computing and ambient intelligence are still relatively new, but gaining more attention. For example, the AutoHAN networking and software architecture [Blackwell & Hague, 2001] supports people to define end-user programmable specifications of interactions between appliances in a domestic environment and describes a tangible interface (Media Cube) for direct manipulation. Truong's CAMP project [Truong, Huang & Abowd, 2004] uses a fridge magnet metaphor that allows users to create context-aware applications for the home by combining words from a controlled vocabulary into sentences to specify behavior. Humble [Humble, Crabtree & Hemmings, 2003] describes a way for end-users to configure their ubiquitous domestic environments using an editor that discovers available ubiquitous components and presents these components as jigsaw pieces that can be combined into applications.

1.2 Smart Retail Environments

Most research on end-user programming of ubiquitous computing environments has focussed on the home domain, like most research on ambient intelligence. New technologies are however seldom first adopted by the home domain. Consider telephony for example. The first telephones appeared in public telephone cells and office buildings, only later did they find their ways into the homes of people and more recently in people's pockets and bags.

Other examples are light bulbs, television screens, computers, printers. The domestication of ambient intelligence may follow a similar route and start first in the professional domain in areas such as monitoring hospital patients, intelligent building control (energy management, security) and traffic control systems. In this thesis we concentrate on intelligent environments in stores and brand spaces which provide entertainment, information or other immersive experiences that enhance the retail function. There are a number of reasons for looking at smart immersive retail environments.

First, shopping has become an important leisure time activity. People do not just go to shops to buy things, they also go to shops and shopping malls for the experience of finding out new things and spending time with friends. Large shopping malls and shopping districts have transformed into tourist destinations that attract thousands of visitors. South China Mall, the largest shopping mall of the world in 2007 [Riper, 2007] had 1,500 stores in approximately 600 square kilometers of total floor area with 7 separate themed zones modelled after cities and places such as Amsterdam, Paris, Rome, Egypt, the Caribbean, and California.

Second, stores have a unique, personal brand identity. Brand image plays an important role in attracting customers and setting the store apart from the competition [Kozinets, Sherry, DeBerry-Spence & Duhacheck, 2002]. In an experience economy where consumers look for authenticity in products and services, and make decisions based on whether an experience is perceived as real or fake, the physical setting can play an important role in rendering authenticity [Pine & Gillmore, 2007]: Branded spaces give companies a chance to show customers what they really are and give customers the choice to experience the identity of a brand and its products and services for themselves and make their own value judgements. Companies such as Apple, IKEA, Volkswagen, Nike have all created branded spaces where people can experience and interact with their brands.

Intelligent environments offer large retailers possibilities to create personalized connections with customers, connect to more customers (using presence in virtual worlds), and respond more rapidly to changes in fashion trends [IBM Retail, 2008] beyond opportunities of becoming tourist destinations and showing who they are to customers in search of meaningful and authentic experiences. Immersive intelligent retail environments can also facilitate potential customers to test-drive a product or service before they buy it as discussed by [Edvardsson, Enquist & Johnston, 2005]. IKEA furniture stores consist of several *experience rooms* where people can try out furniture in a kitchen, living room or bedroom setting for example.

Traditionally, the store frontend has been an important element in the retail environment. The goal of shop windows is to create enough stopping power so that people walking past the store become curious to see what is inside. Furthermore, the shop window is the first place where people can experience what the store or brand stands for and therefore plays a key role in communicating the type and style of the shop to people. The shop window also serves to inform people about the products available. The store frontend is therefore an interesting area to consider for improving with ambient technology and media. One of the first examples is provided by Ralph Lauren who installed a number of interactive intelligent shop windows in their flagship stores in New York, London and Chicago. Customers can browse their collection 24 hours a day and add items to their digital shopping cart [Toler, 2007]. Shoppers are then contacted by e-mail or phone the next day to securely enter their payment information and arrange for shipping. Interactive (transparent) screens can also be used inside the store to provide in-store communication or promote individual brands. In the case of the *Kijkshop*, a Dutch electronics and household appliances retailer, the entire shop is one large shop window customers can walk through.

1.3 Focus of the thesis

This thesis addresses the problem of deriving an organizing concept that captures the way in which we form experiences in our everyday life and integrates interactive ambient media into this process. It proposes the design of a system that supports retailers and designers to create, deploy and maintain their own smart retail environments described in this general interaction concept to support a mass customization strategy towards ambient intelligence. An implementation and user evaluation of such a system is described for intelligent shop windows that react to the presence and activity of people, which can be seen a subclass of smart retail environments.

1.3.1 Research questions

Several ways to design an end-user software engineering approach for intelligent environments exist. The most straightforward way is a bottom-up approach in which individual use-case scenarios of intelligent environments are analyzed for commonalities and differences. From this analysis a general template or prototype scenario would then be reconstructed that can be customized by end-users on different aspects. Although this scenario-based approach [Carroll, 1995] is useful in well-defined and focussed application domains, it is difficult to apply to broader application domains as there seems to be an almost endless variety of use-case scenarios. This thesis proposes a solution approach that starts out in a top-down fashion by deriving a general interaction concept that describes the complex and dynamic nature of physical environments where multiple actors (both human and mediated) may be involved in several activities at the same time using multiple devices simulatenously and that integrates ambient technology and media in this picture. Since it has proven technologically difficult to design a device or system in advance that is capable of generating this intelligence automatically for every individual context of use and economically infeasible to design a tailor-made solutions for each individual situation, this organizing interaction concept needs to support a mass customization approach. Our hypothesis is that such an approach is not only technologically feasible for a large subset of smart retail environments but also suitable and usable for this target user group in their everyday life practice. The first question can then be stated as follows:

1. How can we represent generic ambient intelligent environments in a way that facilitates mass customization?

This representation that we will refer to as an *ambient narrative* can be seen as a high-level description of how ambient intelligence emerges through the situated actions of actors in their environment. To generate more specific functional requirements and arrive at a suitable machine readable representation for such ambient narratives an application domain must be chosen. Because we want retail stakeholders involved in store design to be able to write ambient intelligent environments in this high-level interaction concept we formulate a second question:

2. Can designers in the retail domain understand the ambient narrative concept and what are the specific requirements these users place on a system design?

These different user, application and system requirements can then be translated into a suitable system architecture. As total cost of ownership is a key driver in the retail business, this system architecture should take into account the entire lifecycle of designing, testing, deploying and maintaining retail ambient narratives. At the same time there should be no noticeable difference for the retailer in terms of performance with a custom-built application with exactly the same functionality. We restrict ourselves to a smaller but representative set of smart retail environments, i.e. intelligent shop windows that react to the presence and activity of people. The third question that arises then is:

3. How does a system to control the lifecycle of such an ambient intelligent shop window environment look like?

This system architecture can then be implemented in a prototype system that needs to be evaluated with real users in order to address the fourth and final research question:

4. Can retail designers create ambient intelligent shop windows using the prototype system?

1.4 Contributions

The research described in this thesis has resulted in four main contributions: The ambient narrative concept, its suitability for smart retail environments, a formal model to specify ambient narratives in smart retail environments, and a prototype implementation validated with retail designers. More specifically:

- ◇ The ambient narrative concept as an organizing concept for representing ambient intelligence environments in a way that lends itself for mass customization.

Definitions of ambient narrative and ambient intelligence are given in section 2.3.3. In addition this section shows how this high-level interaction concept for intelligent environments facilitates mass customization of ambient intelligence. Section 2.4 provides different ways to classify ambient narratives to illustrate the applicability of the concept in a wide variety of application domains including retail.

- ◇ Validation of the ambient narrative concept with retail design stakeholders and identification of user, system and application requirements.

Section 3.2 describes the results of a series of workshops and interviews conducted with retailers, consultants and designers to create user-generated ambient narrative examples and test whether retail users can understand the ambient narrative concept (second research question). Section 3.5 provides a list of the functional requirements that were derived from these workshops and interviews. These requirements served as the basis for the design of the prototype ambient narrative shop window system in Chapters 4, 5 and 6.

- ◇ A hypertext language model and formal system architecture to describe and control the lifecycle of intelligent shop window ambient narratives.

Chapter 4 provides a formalization of the ambient narrative concept that can be written down in a hypertext language model to represent ambient narratives. Chapter 5 describes the ambient narrative engine, the core component in the system architecture discussed in Chapter 6.

◇ User evaluation of the prototype ambient narrative shop window system.

Section 7.1 describes the results of a user evaluation study to address the fourth research question. One of the questions left unwritten is to what extent this prototype ambient narrative system can be applied to other application domains than the intelligent shop window. This question is briefly discussed in section 7.3. We say here now that this research should be seen as step towards achieving that goal.

1.5 Outline thesis

The remainder of this thesis is organized as follows. To approach the research questions stated before, we adopt the spiral model for software development (see [Boehm, 1988] and Figure 1.1 for a simplified version). In total we follow two iterations: The first iteration is described by Chapters 2 and 3 and is conceptual. The second iteration is covered by Chapters 4 to 7 and forms a first design iteration.

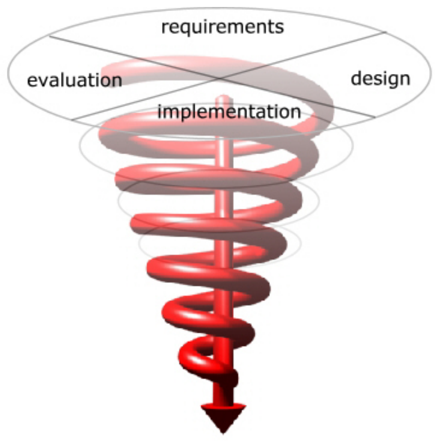


Figure 1.1: spiral model

Chapter 2 investigates the social-cultural and economical factors that shape ambient intelligence. In particular we look at the central role performance plays in our everyday life and the emergence of the experience society that draws cultural performance into the social-economical sphere. This background is used to derive the ambient narrative concept and present a taxonomy of ambient narratives to discuss related work.

Chapter 3 places this ambient narrative concept in a retail context to evaluate whether the target user group can understand and work with this concept. We provide and analyse a list of user, system and application requirements derived from interviews and workshops with retailers, designers and retail consultants and an additional literature study on application user scenarios. Chapter 3 also introduces the intelligent shop window application as the subclass of smart retail environments and presents the results of a user evaluation on different authoring strategies.

Chapter 4 translates the informal requirements of Chapter 3 that relate to the ambient narrative concept and the run-time behavior of the sytem into a formal model. The problem of mass customization of ambient intelligence from a set of dynamically changing modular fragments is formally described and explained. Related work in situated action modelling, action selection as interactive storytelling and authoring is also discussed.

Chapter 5 provides a detailed implementation of this formal model in the ambient narrative engine that forms the central component in the shop window system architecture.

Chapter 6 discusses the remaining informal requirements of Chapter 3 that refer to managing the lifecycle of ambient narratives and uses these requirements to make design decisions that are implemented by the intelligent shop window system architecture and its individual components (except for the ambient narrative engine component).

Chapter 7 evaluates the programmability of the intelligent shop window system by end-users by means of a user study and technical analysis. It also discusses the limitations we posed upon ourselves by choosing this subclass of smart retail environments.

Chapter 8 presents the conclusions and discusses directions for future work.

2 Ambient Narrative Concept

In this chapter we follow a top-down analytical approach to derive a concept that captures both the variety and dynamics in ambient intelligent environments in a way that facilitates mass customization. First we need to understand better what ambient intelligence is and how it supports people in performing their everyday life activities. In section 2.1 we start by looking at the central role of play and performance in our culture and society. In particular we are interested in the everyday life performances that we enact when we e.g. wake up in the morning, brush our teeth, have breakfast and drive to work and the role of technology in enhancing these activities. In section 2.2 we investigate how this view of social life as theatre is also reflected in an experience economy where we pay for access to commodified cultural experience. Section 2.3 applies literary semiotics theory to describe the interaction between people and environment in which they perform their social cultural scripts on a more abstract level in order to derive the ambient narrative concept for representing intelligent environments. In section 2.4 we present two genre taxonomies in ambient narratives to discuss and structure related work.

2.1 Everyday life performances

All the world's a stage, And all the men and women merely players; They have their exits and their entrances; And one man in his time plays many parts, His *acts being seven ages.*– William Shakespeare, As You Like It (II, vii, 139-143)

The goal of ambient intelligence is to help people performing their daily activities better by making these activities more convenient and enjoyable through the use of interactive media. Notice the word *performing* in this description. It is a word used so commonly in our language that we forget that performance is pervasive in culture and society, not just in the theatre, music, dance and performance arts but also in our everyday lives: We perform the role of a father or son in our private lives but maybe also that of a doctor, judge, or police officer, for example in our working lives.

Because performances vary so widely from medium to medium and culture to culture, it is difficult to pin down an exact definition for performance. Richard Schechner defines performance as "ritualized behavior conditioned/permeated by play" or "twice-behaved behavior" [Schechner, 2002]. When people are performing, they display behavior that is at least practiced once before in a similar manner. In traditional performance arts this behavior can be detected easily: Actors in a theatre play, opera, or movie rehearse their roles offstage and repeat this behavior when they are onstage. But this twice-behaved behavior can also be seen in a priest conducting a wedding ceremony, a surgeon operating on a patient, or a fastfood service employee behind the counter. Even in our own homes, people show signs of this repeated behavior. This happens for example during everyday rituals, like watching a soccer match with friends, end of the day routines after coming home from work in the evening.

Although many different perspectives on performance studies exist, they agree that people follow culturally specified social scripts that influence each other [Goffman, 1959; Burke, 1966; Turner, 1975; Fisher, 1989]. These everyday life performance are much less scripted than theatrical performances and several authors view social interaction as an improvised performance [Sawyer, 2001; Boje, Luhman & Cunliffe, 2003; Grove, Fisk & LaForge, 2004; John, Grove & Fisk, 2006]. However, Erving Goffman and others also emphasize that our everyday life performances are highly constrained at the same time by the scripts that we learn from our culture and that we are less spontaneous than we would like to think.

The dramaturgical theory of social life is relevant to ambient intelligence for two reasons: First, if people behave according to (improvised) social scripts, we may succeed in codifying interactive media applications to support people in carrying out these scripts. Just as lighting and sound effects add to the overall drama of a theatre play, ambient intelligence may be applied to improve the setting of the social front, support dynamic realization or avoid misrepresentation for example. Second, positioning ambient intelligence in this dramaturgical theory of social life opens up a well-defined and familiar frame of reference for the design of ambient intelligence environments and the underlying technology.

Figure 2.1 shows an example of how ambient media can improve the physical setting of waking up in the morning. Instead of a buzzing alarm clock that goes off in the morning, the Philips Wake Up Light simulates a sunrise with (optionally) the sound of singing birds, seashore or forest pond animals to create a more natural wake up experience [Philips, 2007].

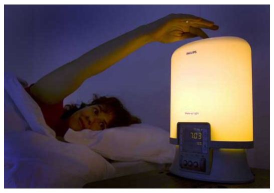


Figure 2.1: Supporting the wakeup experience with light and sound

Figure 2.2 shows a cartoon projected on a mirror TV (a one-way mirror with an LCD screen behind) that invites the small child standing in front of the mirror to brush his teeth for two minutes. The cartoon carries the child through this daily task and helps the child to maintain expressive control.

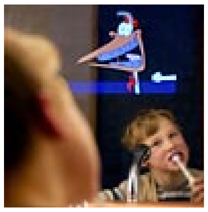


Figure 2.2: Enhancing the toothbrushing performance with graphics

Another way of looking at this example is that the cartoon shifts the attention from achieving the end result to the process of getting there and brings the child in a flow experience [Csikszentmihalyi, 1990] where the child is fully immersed and engaged in performing the task. Many children (and adults) like the demonstration in ExperienceLab [Lazeroms, 2002] and want to play along but it has yet to be proven whether the effect is also positive in the long term when the novelty of the cartoon wears off and people start to try out different things (e.g., what happens if I do not brush my teeth?). At first glance there seems to be no audience present. But if we look closer at Figure 2.2 we see that the child is performing his act, the toothbrush activity, in front of the cartoon character. The cartoon character improvises its performance based upon the actions of the child. Goffman's dramaturgical theory of social life extends to computers. In fact it turns out that people are treating computer characters as real people and see these characters as an audience that watches their everyday life performances. In their book *The Media Equation* [Reeves & Nass, 1996] present a series of psychological experiments that show that people are polite to computers, react differently to male and female voices, and react to large faces on a screen as if they are live people invading their personal body space, for example. The "media equation" explains why we have higher expectations that a human avatar will behave intelligently than, for example, an animal-like virtual character. This idea of viewing human computer interaction as theater can be carried even further in the design of human computer interfaces [Laurel, 1986]. Rather than viewing the computer as a tool, Brenda Laurel views the computer as a medium for interaction in which the user (audience) becomes an actor, active entity on the stage, the virtual world that is populated by agents, both human and computer-generated and other elements (props) of the representational context (windows, desktops).

2.2 Experience economy

Tell me and I will forget, show me and I will remember, involve me and I will understand – Confucius

The dramaturgical theory of social life has been adopted by organization studies and service marketing literature [Grove & Fisk, 1983; Fisk & Grove, 1992; Harris, Harris & Baron, 2003; Stuart & Tax, 2004; Oswick, Keenoy & Grant, 2001; Clark & Mangham, 2004]. In service marketing, the service encounter can be seen as a performance that takes place on a particular location (stage) and that involves service employees (actors) that deliver the service to customers (audience). The service manager as the director of the service performance is the person responsible for carrying out the business strategy (script). Employees practice (rehearse) this script back stage before they go on (front) stage [Fisk & Grove, 1992; Pine & Gillmore, 1999]. Managers cast themselves into charismatic roles while employees are casted as allies to manage the impression on the audience [Gardner & Avolio, 1998].

Within organization studies and service marketing there is a growing body of research that not only accepts the organizational theatre as a metaphor but take it literally [Pine & Gillmore, 1999; Clark & Mangham, 2004; Boje, Adler & Black, 2005]. Joseph Pine and James Gillmore describe an experience economy in which "work is theatre and every business a stage" [Pine & Gillmore, 1999]. They argue that staged experiences are valued higher by customers than delivered services, manufactured goods, or extracted raw materials. Experiences fulfil a larger subset of customer needs, so customers are willing to pay a higher price. Jeremy Rifkin further argues that in this experience economy, culture itself is being pulled into the commercial sphere. Entertainment parks, themed shopping malls, and organized tourist travel are just some examples where people no longer buy things, but pay to gain access to services and commoditized culture [Rifkin, 2001].

Empirical evidence suggests that the 'servicescape', the environment of the service, plays an important role in how people perceive a service encounter [Bitner, 1992]. This suggests that ambient intelligence can also be applied in professional service encounters to enhance the atmospherics of a service encounter and thereby the perception or performance of a service in a positive way. Consider for example a medical imaging room in a hospital. Many patients feel frightened by the bulky equipment. By enhancing this environment with immersive media for example, projecting personalized video clips that are selected by each patient on the walls and ceiling of the examination room-patients may feel more at ease, as illustraed in Figure 2.3.

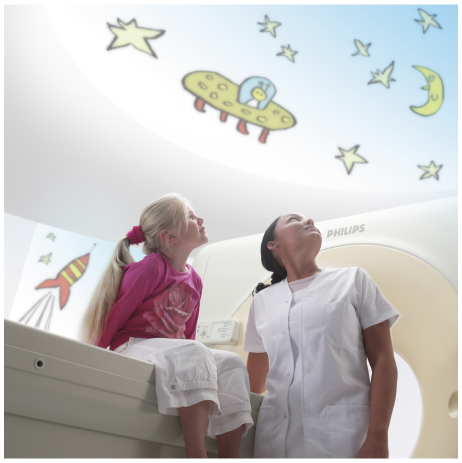


Figure 2.3: Improving the performance of medical examination with ambient media

Philips demonstrated this application concept in the ambient experience pavilion at the annual meeting of the Radiological Society of North America (RSNA) in December 2003. Today, several installations are being used in hospitals around the world. In practice, about one-third of the children scanned with a traditional scanner require sedation because they are unable to relax enough for a successful diagnostic exam. This can add several hours of recovery time to a procedure that could be completed in 15 minutes [Physorg, 2007]. It turns out that patients feel less anxious with this ambient experience than without it and this results in a lower percentage of people, especially children, who need sedation and therefore less time these people have to spend in the hospital.

As people increasingly make consumer choices based on the symbolic value of consumer goods and with the goal to affect their own inner life [Baudrillard, 1985; Baudrillard, 1995; Schulze, 1992], every aspect surrounding the purchase of a product, use of a product or delivery of a service can be embraced by companies to infuse it with meaning. This can be the design of the consumer good, the values of the company carried in the brand name but also the physical environment where the product can be experienced. Themed restaurants and hotels have been around for some time [Gottdiener, 2001; Sherry, 1998], more recently brands such as Heineken, Volkswagen, Apple, Nokia have opened up brand centres and flagship stores [Kozinets, Sherry, DeBerry-Spence & Duhacheck, 2002] with a strong experiential environment in many locations around the world, sometimes with great success: Revenue for each square foot at Apple stores in 2005 was $2,489, compared with $971 at Best Buy, a big computer and electronics retailer in the same year [Lohr, 2006]. In the first quarter of 2007 more than 21.5 million people visited the more than 180 Apple stores. Store sales were $855 million and they contributed more than $200 million in profits [Stross, 2007].

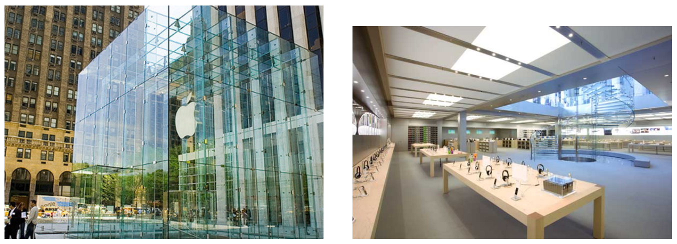


Figure 2.4: Apple flagship store on Fifth Avenue, New York

More recently, several authors [Zuboff & Maxim, 2002; Prahalad & Ramaswamy, 2004; Boswijk, Thijssen & Peelen, 2007] describe a trend away from staged (themed) and mass customized experiences towards a different model where the experience is co-created in a continuous dialogue between the company and the individual. By allowing individuals an active role in the construction of the experience, the level of agency, of being part of the event is increased, leading to a richer, more meaningful and memorable experience. This active role in value creation is different from allowing customers access to the technology base of the company or having a customer interface that enables customers to choose which features must be included in the final product as in a mass customization strategy [Davis, 1987; Pine, 1992]. It is also different from user-centric or participatory design or the use of anthropologists because although potential customers are involved in the (concept) design phase and early stages of product development, the dialogue is discontinued at the moment the product is mass-produced. In all these methods, the firm creates the value and the focus of innovation is on product and processes whereas in the experience co-creation model value is co-created and the focus of innovation is the experience environment itself. An example of such an experience environment is Lego Factory [Lego, 2008]. On the Lego website, people can download Lego Designer, a PC application that enables Lego fans to design their own Lego models. When they are happy with the result, they can upload their model to the Lego Factory site. Lego calculates which Lego bricks are needed, assembles the Lego bricks in a custom-made Lego box and finally ships it to the consumer. The Lego community website furthermore gives fans the possibility to share models with each other. The gravity of innovation in Lego Factory shifts away from the Lego organization to the Lego community, Lego Factory establishes itself as the nodal company in the network that supports the individuals. Second Life [Linden Labs, 2008], a virtual online world where players actively participate to the build the virtual world and Vocation Vacations [Vocation Vacations, 2008] that offers people the choice to test drive their dream job with the help of a mentor are other examples.

2.3 Ambient narratives

Our story is in the land ... it is written in those sacred places ... My Children will look after those places, that's the law. – Bill Neidjie, Kakadu elder

Before we can derive a suitable abstract representation for intelligent environments (section 2.3.3) that captures the variety and dynamics of everyday environments, it is necessary to analyze how personal experience and meaning is being formed as people perform their social scripts in experience co-creation environments. This process can be seen as a process of manipulating symbols and since computers are symbol processing systems and our goal is to find a representation that is readable by computers in Chapter 3 we adopt a literary semiotical approach [Chandler, 2001; Rivkin & Ryan, 1998] to derive a representation for intelligent environments. Broadly speaking, experience co-creation environments are characterized by high levels of immersion and interactivity. First the immersion dimension will be discussed through a semiotic perspective (section 2.3.1), then the interactivity dimension (2.3.2).

2.3.1 Spatial storytelling

In section 2.1 the focus was on performance and play as the object of study and not so much on the environment where the performance is situated. By performing in our environment we can choose the situation we are in but this situation will affect, alter us in a way we cannot control: Experiences are formed in a subject-determined, reflexive and involuntary way [Schulze, 1992]. When we form experiences, we read the signs (i.e. units of meaning) encoded in our environment as text and create a personal meaning or story in our mind. Text is therefore not restricted to written words on a piece of paper or characters on a computer display.

Aboriginal culture for example views the landscape of Central Australia as a text that is to be preserved for future generations by the people of Australia: Aboriginal creation myths speak of legendary metaphysical beings that wandered the earth during the Dreamtime, singing out the names of everything that crossed their paths and in doing so they sang the world into existence. In the Aboriginal worldview, every meaningful activity, every event that occurs at a particular place leaves behind a vibration in the earth that echoes the events that brought that place into existence. The Aboriginals see themselves as the preservers of the invisible pathways or songlines that define Australia and take great care not to disrupt these pathways by their actions [Chatwin, 1988].

Stories are not only found in the landscape, they can also be found in architecture. Our perception of architecture is a function of the properties of the enclosed space, cultural background and prior experience [Ching, 1996]: As we walk through a building, we read the story that is being encoded in these properties. If this story resonates with us, the piece of architecture becomes meaningful to us, or as Le Corbusier writes: “Architecture is a thing of art, a phenomenon of the emotions, lying outside questions of construction and beyond them. The purpose of construction is to make things hold together; of architecture to move us. Architectural emotion exists when the work rings within us in tune with a universe whose laws we obey, recognize and respect. When certain harmonies have been attained, the work captures us.” [Le Corbusier, 1960] (p.23). Throughout history, many visual spatial storytelling environments have been created for various purposes; see [Grau, 2003] for an overview and discussion.

Theme parks explicitly create strong narrative structures in the landscape, the architecture, the cast and the attractions people visit. Don Carson, imagineer at Disney and video game designer, describes that “one of the secrets behind the design of themed environments is that the story element is infused into the physical spaces as the guest walks or rides through" and "that every texture, every sound, every turn in the road should reinforce the concept" [Carson, 2000]. Installation art [De Oliveira, Oxley & Petry, 2004] is another example of such spatial storytelling environments. Installation art has its origins in the *Gesamtkunstwerk* idea of music composer Richard Wagner. Wagner envisioned the integration of different art forms, stage performance, visual art and music art in what he called a Gesamtkunstwerk. Installation art is not defined or classified in terms of a traditional medium, but in terms of the message it conveys by whatever means.

As we perform, e.g. walk, drive or swim through such spatial storytelling environments we use all our senses simultaneously to read the text that is encoded in various ways (e.g. marks in the landscape, architectural properties, physical objects) and carried by multiple modalities (e.g. audio, visual, haptic). Because our senses work in parallel (and each have a limited bandwidth) we can therefore read more text in the same amount of time as in the case of e.g. reading a book or watching a movie. The more text can be read in a certain amount of time, the higher the chance of being immersed into this text, provided the message of the text is internally consistent. User experiments confirm that a movie viewed with *AmbiLight* [Diederiks & Hoonhout, 2007] is seen as more immersive than as one without. Random ambient lighting on the other hand destroys the sense of immersion in the movie.

2.3.2 Reader text interaction

If environments where experiences are staged, personalized or co-created can be seen as texts that are read, it is possible to set the perceptual characteristics of such environments aside and focus on the interaction between the reader (performer) and this text.

An experiential service that is being staged in front of an audience forms a narrative in the minds of people. Theories of narrative and in particular Russian formalism and French structuralism have made a distinction between what is told (the story) and how it is told (discourse) [Chatman, 1978]. A third category is the text consisting of signs that is produced by an agent who conveys the story as discussed in 2.3.1. The story-part can be split into two parts, existents (actors and settings) and events (actions and happenings). We learn the story via discourse so the way the story is told can have a strong influence on how the story is being perceived. Discourse analysis is concerned with aspects such as plot (the causal and logical structure which connects events), narrative voice (who tells the story), point of view (who sees the story) and narrative modes (mimesis vs. diegesis or showing vs. telling). The story of a staged experience is then the sequence of events as they happened on stage. The narrative voice is that of the actors on stage and the point of view is the customer's. The narrative mode is mimetic, there is a direct representation of speech and action. The plot is fixed, customers cannot change, alter or otherwise affect the causal and logical structure of events.

More recently, post-structuralists thinkers like Rene Barthes, Michael Foucault, Jacques Derrida and Julia Kristeva reject the belief in symbols of constant and universal significance upon which structuralism rested, see e.g. [Rivkin & Ryan, 1998]. Because of this unstable meaning and absence of any objective truth, the meaning intended by the author of a text is seen as secondary to the meaning the reader perceives. The literary critic should 'deconstruct' the text according to Derrida to create a multifaceted interpretation of the text using a variety of different perspectives. This for example also includes analyzing why certain words have not been written as this can reveal dominant relations in the text. Kristeva coined the notion of intertextuality to indicate that texts are not isolated from each other but influence and shape each other in different ways.

Reader-response theorists (see e.g. [Martin, 1986]) such as Rene Barthes, Jonathan Culler and Wolfgang Iser join this line of thinking by arguing that meaning comes into existence through the act of reading itself and is not something that is hidden in the text waiting to be found. Reader-response theorists view literature as a performing art in which each reader creates his or her own text-related performance. The reader is always looking both backward as well as forward in the text, actively structuring each part of new information encountered. Iser shows how abstractions such as implied readers are mobilized and altered in their journey through a story and how the expectations, meanings and beliefs about characters and settings are explicitly changed during the act of reading. Barthes in his post-structuralist analysis of the short story Sarrassine van Balzac proposes five 'codes' (hermeneutic code, code of semes, referential or cultural code, code of actions or proairetic code, and code of symbols) and shows how they create multiple possibilities of meaning as the readers follow them through the text.

Although customers as readers of a staged experience may each assign different meanings to the text they encounter, the story presented is the same for all customers. In mass customization environments readers can choose which components should be assembled to create a personal experience, story. This choice of components can be explicit like clicking on a hypertext link or more implicit: Depending on the way we perform, e.g. walk through a park or museum or look at painting, a different is story is revealed to us. A mass customization environment can therefore be seen as interactive narrative defined by [Meadows, 2002] as a "time-based representation of character and action in which a reader can affect, choose or change the plot. The first-, second-, or third person characters may actually be the reader. Opinion and perspective are inherent. Image is not necessary, but likely." Hypertext novels and other forms of interactive fiction such as computer adventure games are often classified as interactive narrative [Laurel, 1993; Murray, 1998; Meadows, 2002] and contrasted with more traditional media such as books or movies, but the division may not be so clear [Aarseth, 1997]. The author of a book can design the chapter layout for a book in such a way that it supports different reading paths similar to the author of a hypertext.

To analyze the role of the author and reader in interactive narrratives it is more helpful to view narrative as a communication between a sender and receiver, author and reader as depicted in Figure 2.5.

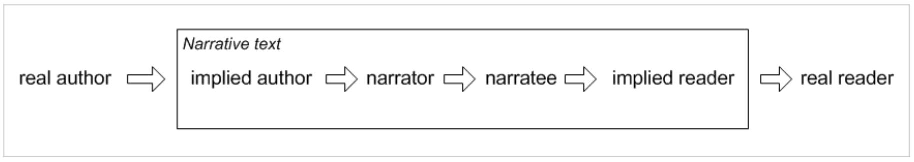


Figure 2.5: Model of narrative communication

In contrast to classical narratives there is a discontinuity between the author and implied author (the author's second self inside the text) and the implied reader (the audience presumed to be present inside the text) and the reader. The author of the interactive narrative has deconstructed the narrative into modular parts each having an implied author, narrator (one who tells the story), narratee (one who listens to the story) and implied reader that make up a plot structure which can be explored by readers. The reader determines which fragment the implied reader will experience next through interaction. The challenge of writing interactive narratives is to balance story and interactivity [Bates, Loyall & Reilly, 1991; Galyean, 1995; Meadows, 2002] to create a coherent story no matter which reading path is taken.

In many computer games and other forms of electronic cybertext there is no narrated plot that is under control of the reader. Instead the story is formed as the result of an interaction between the user(s) and a symbol processing system (machine or human), which may generate a different semiotic sequence each time it is accessed. Espen Aarseth uses the term *ergodic discourse* [Aarseth, 1997] to describe this type of discourse that involves a nontrivial amount of work (*ergos* in Greek) of the reader. Furthermore Aarseth introduces a number of variables with different possible values that enable him to define a typology for characterize the textual dynamics present in different forms of (electronic) literature that is helpful to characterize experience co-creation environments. Table 2.1 sums up these variables and values:

Variable	Possible value
Dynamics	Static, IDT, TDT
Determinability	Determinable, indeterminable
Transiency	Transient, intransient
Perspective	Personal, impersonal
Access	Random, controlled
Linking	Explicit, conditional, none
User function	Explorative, configurative, interpretative, textonic.

Table 2.1: Terminology for (electronic) texts

Dynamics refers to how the text is revealed to readers. A distinction can be made between the sequence of symbols as they appear to readers (scriptons) and the sequence of symbols as they appear in the text (textons). A static text has both fixed scriptons and textons whereas a dynamic text either has variable scriptons and fixed textons (intratextonic dynamics) or both variable scriptons and textons (textonic dynamics). In addition to the notion of scriptons and textons, a text has a traversal function, which he defines as the "mechanism by which scriptons are revealed or generated from textons and presented to the user of the text" [Aarseth, 1997]. An interactive narrative where readers are unable to change the modular story fragments can then be characterized by intratextonic dynamics with an explorative traversal function (reader must decide which path to take). Experience co-creation environments have a textonic traversal function and dynamics (reader can add/remove textons).

Although Aarseth's typology sheds more light on the differences between classic texts, hypertexts and computer games, it only considers the surface text, i.e. the text that appears to the user. Cybertexts can however have different layers of text and readers. The high complexity and associated development costs of modern computer games [Blow, 2004] has led many game studios to reuse the core game engine for multiple games or license it to other game developers, see e.g. [DevMaster, 2008] for a list of commercial and open-source game engines. In this case the game engine text (source code) is read by a language compiler and compiled into the game engine executable. This game engine reads the game level descriptions and creates the simulation. This game level text can be modified by the reader, author or the game itself and this can take place before, during or after the game. Many computer games have a level editor that allows players to create new levels they can play later. In massive multiplayer online games (MMORPG) and alternate reality games [Szulborski, 2005] both players and professional game developers alter the game level (content) as the simulation progresses [Tychsen, Hitchens, Brolund & Kavakli, 2005]. Other games employ procedural generation techniques to generate game level content on the fly or mix procedural generation with user-generated content [Kosak, 2005]. These and other examples show that the separation between reading and writing increasingly blurs in these (virtual) experience co-creation environments and eventually disappears: In a never-ending loop, the performance of the reader rewrites the simulation and the simulation rewrites the performance of the reader. Being (narrative) and becoming (simulation) become two different ways of looking at the temporal consistency of a situation [Cameron, 1995]: If time stands still, the situation appears to us more as a narrative: We are thrown out of the situation and from a distance we can read the text as it exists at that moment and form an impression in our mind. If time passes, the situation appears to us more as a simulation: Through our performance in the cybertext we can change the course of events and automatically feel as being part of the cybertext as it becomes.

2.3.3 Definitions

If the environment can be seen as a text that is read using multiple senses (2.3.1) by people who simultaneously improvise culturally defined social scripts that affect, alter or otherwise changes this text again (2.3.2), we can define an **ambient narrative** as a *spatial-temporal multi-sensory interactive narrative consisting of interrelated media-enhanced social script fragments that are simultaneously read and rewritten by the performance of actor(s) (human or machine)*

The word spatial-temporal indicates that ambient narratives are not virtual environments but environments grounded in real space and time. This space can be identical to a particular place but can also be a union of a set of real spaces distributed in space and/or time such as the individual stores of a retail chain or the concert stage of a rock band touring the world. As people explore these environments multiple senses are used simultaneously to read the ambient narrative (multi-sensory). Media-enhanced refers to the presence of digital media and devices in the environment to support the performance of everyday life social scripts. A place without technology embedded in the surroundings falls outside the scope of the definition. Furthermore, these media-enhanced social scripts are related: Depending on whether a person moves into the bathroom or living room a different set of social scripts will become active or deactive and this will cause particular devices, media and/or ambient effects to start or stop. The collective performance of all the actors will affect how the story progresses and may at the same time also alter the interactive narrative itself. Figure 2.6 clusters different types of texts along the dimensions of multimodal immersion (2.3.1) and interactivity/participation (2.3.2). The texts in the upper part (and in particular the upper right) can be seen as ambient narratives as they have both a multi-sensory, spatial aspect and a dynamic textual component.

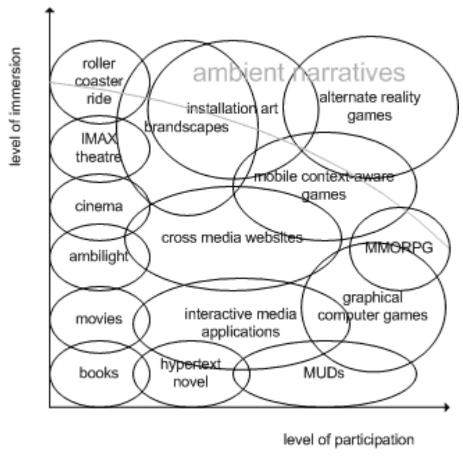


Figure 2.6: immersion vs. participation

With this definition of ambient narrative in place it is now possible we define **ambient intelligence** as *the part of the emerging story in an ambient narrative as the result of the collective performance of actors that is conveyed through the set of user interfaces of the devices surrounding the actors in the environment.*

The ambient narrative definition inherently implements a mass customization strategy because by performing in their environment people assemble, personalize their own story or ambient intelligent experience from the available parts, i.e. media-enhanced social scripts present in the ambient narrative. The media-enhanced social scripts are however not likely to be written in advance by the technology provider of the ambient narrative but by its users because the social scripts differ from place to place and time to time [Brand, 1995] and therefore also the media and ambient effects attach to them. Moreover, the social scripts may be improvised by people and change the ambient narrative at run-time instead of design-time. Note that this does not imply there is no mass customization taking place; at the core of these co-creation environments there can still be a mass customization process, the underlying set of modular parts and their interrelationships is just dynamic. End-user programming environments form a specific subset of co-creation environments as they make people consciously aware that they are creating or modifying an application by providing an interface to visualize and write program behavior.

2.4 Genre taxonomy

To conclude this chapter we present two different ways to classify ambient narratives with the goal to discuss and position related work.

2.4.1 Type of experience

One way to organize ambient narratives is by looking at the type of experience. Pine and Gilmore define four types of experience: educational, aesthetic, entertainment and escapist [Pine & Gillmore, 1999] (and combinations thereof). These types of experiences vary in terms of active vs. passive involvement and absorption vs. immersion as shown in Figure 2.7. Absorption is defined as "occupying a person's attention by bringing the experience into the mind", immersion as "becoming physically (or virtually) a part of the experience itself" [Pine & Gillmore, 1999] (p.31). The involvement dimension is orthogonal to the participation dimension in Figure 2.6.

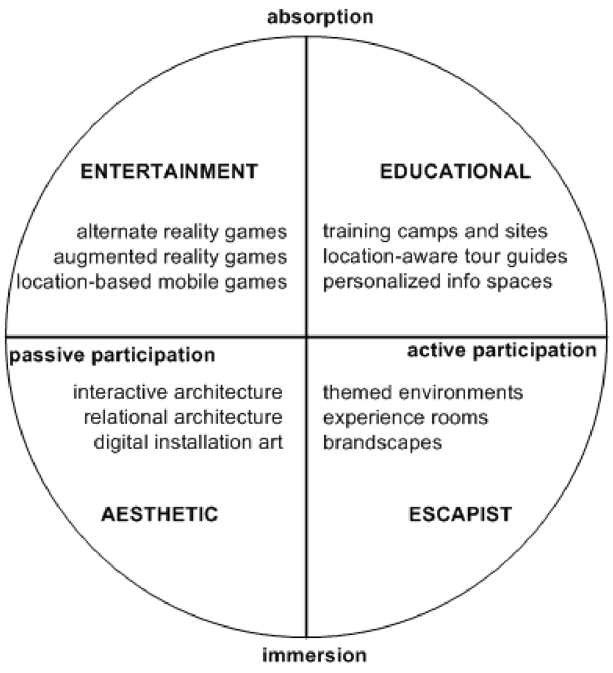


Figure 2.7: Genres in ambient narratives organized by types of experience

Examples of aesthetic ambient narratives can be found in digital installation art [Database of Virtual Art, 2008] and architecture that responds to the presence and activity of people [Glynn, 2008]. The *Weather Project* [Eliasson, 2003], Figure 2.8 (left) was an installation in the Tate Modern museum in London that consisted of hundreds of monofrequency lamps on a giant circular disc hanging at the far end of a large hall to represent a sun to create a vast duotone landscape. Spectators could see themselves ithrough the cloud-like formations of fog that was dispersed into the space in mirrors in the ceiling. *Sensing Speaking Space* [Legrady, 2002] Figure 2.8 (middle) and *The Living Room* [Sommerer & Mignonneau, 2001] are some examples of interactive installations that create a space that becomes alive and starts to sense users as they enter and interact with the room. In Sensing Speaking Space an image is projected on a large screen and sounds are spatialized around the audience through a 6 channel surround-sound based on their locations in the gallery. The image and spatial sound changes as people perform in space in front of the installation. The Living Room interprets the actions of people in the environment to retrieve images from the Internet and blends these into the projections around the users. *Vectorial Elevation* [Lozano-Hemmer, 2004] was an interactive artwork that transformed the sky above the city of Dublin, Ireland. Using a 3D interface and a website people could design huge light sculptures that would be rendered with 22 robotic searchlights located in the city in the sky above. *House-swarming* [Didier, Hess & Lutyens, 2007] Figure 2.8 (right) is an installation that operates both as a complex light pattern that greets visitors and as an environment sensing system. During the day, swarms of green ambiguous forms accentuate the entry of a building. At the end of the day the swarm begins to come alive, telling visitors and people passing by about e.g. the air quality around the building.

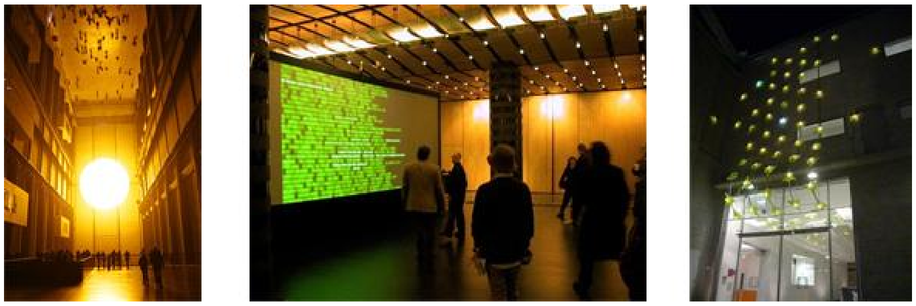


Figure 2.8: Aesthetic/Digital installation art

Location-based games as described by e.g. [Björk, Holopainen, Ljungstrand & Akesson, 2002; Bell, Chalmers, Barkhuus, Hall & Sherwood, 2006; Rashid, Bamford, Coulton & Edwards, 2006; Baron, 2006] use a GPS-enabled mobile phone or other portable device equipped with support for some kind of localization technology to determine the location of players in the real world and change the plot of the story based on their position in the real world and the actions they take. A popular location based game in Japan is *Mogi*

[Newt Games, 2003]. The goal of the game is to pick up virtual items with a mobile phone that are hidden away in real places. Using their mobile or via the website players trade these items to complete item collections and score points. In augmented reality games such as [Thomas, Close, Donoghue, Squires & Bondi, 2002; Cheok, Goh, Liu, Farbiz, Fong & Teo, 2004; Norton & MacIntyre, 2005] users wear a head-up display that superimposes the virtual view on the real-world view in real-time and in 3D dimensions. In *AR-Quake* [Thomas, Close, Donoghue, Squires & Bondi, 2002] (based upon the computer game Quake) players have to find and kill virtual monsters that are hiding in the real world. An augmented reality game where players have to control virtual trains on a real wooden miniature railroad track using a hand-held device that visualizes the invisible trains is described by [Wagner, Pintaric, Ledermann & Schmalstieg, 2005]. In both location-based games and augmented reality games players experience the game through or on a single portable device. More immersive examples of entertainment ambient narratives can be found in alternate reality games [Szulborski, 2005]. Alternate reality games are interactive narratives that are not restricted by medium or platform: The plot can progress through e.g. printed ads in newspapers, websites, phone calls, emails, and live events. Alternate reality games are designed in such a way that players have a meaningful role in creating the story. Players have to work together to solve the game. Because of their open-ended nature, the game designers or *puppetmasters* continue to create new plot material as the game runs. *The Beast* [McGonigal, 2003] is generally considered to be the first mature alternate reality game. It was created as a viral marketing campaign to promote Steven Spielberg's movie *A.I.: Artificial Intelligence*. Other examples of alternate reality games are e.g. *I Love Bees* (2004), *Perplex City* (2005), *Cathy's Book* (2006) and *World without Oil* (2007) [ARGNet, 2008]. An example of an indoor alternate reality game is the *5W!TS Tomb Experience* [5W!TS, 2007] in Boston as shown in Figure 2.9 (left). In groups of 2-15 people and accompanied by a guide, players explore an archaeological dig site to find the pharaoh's burial chamber. The path and story of the 40 minute adventure aren't fixed, but depend on whether participants are able to solve challenges and avoid traps they find along the way. The tomb experience includes sound effects and special effects that include dropping ceilings, trap doors, lasers, fog and shooting air jets. *La Fuga* [McHugh, 2006] in Madrid places players in a high-tech prison from which they have to escape as shown in Figure 2.9 (right).

Context-aware tour guide applications for exploring cities e.g. [Abowd, Atkeson, Hong, Long & Kooper, 1997; Cheverst, Davies, Mitchell & Friday, 2000; Izadi, Fraser, Benford & Flintham, 2002] and museums e.g. [Izadi,

Figure 2.9: Entertainment/Alternate reality games

Fraser, Benford & Flintham, 2002; Bowen & Fantoni, 2004; Chu, 2006] that personalize information presented to users based on their knowledge level or prior experience are examples of ambient narratives that are designed with the goal to support learning. Similar to location-based games the story is conveyed by means of text, audio and video on a portable device and depends on the location of the user in the physical environment. An adaptive museum guide that combines education and entertainment is described by [Wakkary & Hatala, 2006]. The e(c)ho system presents sounds such as the sound of animals or their natural habitat based on the location of visitors in the exhibition space. The audio is acoustically more prominent if a nearby artifact matches the visitor's interest. If a visitor is standing directly in front of an object, a playful tangible user interface in the form of a wooden cube can be used to navigate between linked audio objects associated to the displayed object. Location-aware services or information spaces can also be classified as educational ambient narratives. Examples are e.g. public ambient displays [Vogel & Balakrishnan, 2004; O'Neill, Woodgate & Kostakos, 2004], mobile retail and product annotation systems [Smith, Davenport, Hwa & Combs-Turner, 2004] and public transportation information systems [Repenning & Ioannidou, 2006]. Among the most immersive and engaging educational experiences belong military training camps like the Fort Polk Joint Readiness Training Center in Louisiana (Figure 2.10). This US Army training facility that simulates the Middle East forms the last stop for thousands of soldiers before they are deployed to war conflict situations around the world. Over 400 square kilometers in size and complete with 18 fake Iraqi towns complete with mosques, schools and houses and including 1200 role-players who act as Iraqi civilians, mayors, imams, journalists and humanitarian aid workers, the aim of the JRTC is an ultra-realistic and detailed live simulation to give commanders and soldiers the "experience before they experience it" [Beiser, 2006]. During exercises, fireball cannons simulate explosions and fog machines and speakers fill buildings with smoke and the sounds of gunfire or barking dogs to create a violent ambient intelligent environment.

Figure 2.10: Educational/Training camps and sites

The fourth quadrant is occupied by escapist experiences. Escapist experiences let people 'escape' into an immersive environment where they can just be. Examples include themed restaurants such as Planet Hollywood, Hard Rock Cafe, Rainforest Cafe and the Marton (Chinese for toilet bowl) restaurant in Taiwan [Pescovitz, 2005] and themed hotels like the Venetian in Las Vegas and the Hydropolis underwater hotel in Dubai [Hydropolis, 2008]. Often these environments include (interactive) lighting and ambient effects to strengthen the theme. Another example of escapist ambient narratives can be found in places where customers can experience the brand of a company as discussed in section 2.2. NikeTown London, Nike's flagship store in the UK is a three story building that consists of separate housing areas, each dedicated to a specific sport, which surround a central town square. In the middle of this square is a 360 degree projection screen that displays photos about sports and several interactive displays that reveal background information about Nike products, Figure 2.11 (left). Every twenty minutes this central core comes to life and starts a light and projection show. NikeTown also organizes special events such as interviews with athletes and running clubs to turn the shop into a destination where people enjoy spending time [Klingmann, 2007]. In the Nokia flagship store in Helsinki the lighting, music and content on the screens changes over the course of the day. In front of the screens hanging against the wall, people can pick up Nokia phones and see information about this mobile phone projected over the screen [1] as shown in Figure 2.11 (right).

[1]personal visit, August 2007

Figure 2.11: Escapist/Brandscapes

2.4.2 Service field

Ambient narratives may also be classified by service fields; [Fisk & Tansuhaj, 1985] define ten broad service categories: Health care services (e.g. hospitals), hospitality, travel and tourism services (e.g. hotels, restaurants), financial services (e.g. banks), professional services (e.g. real estate, accounting firms), sports, arts and entertainment services (football, opera, rock concerts), channel, physical distribution, rental and leasing services (e.g. retailing, automobile rentals), educational and research services (e.g. day care, libraries), telecommunication services (e.g. internet cafes), personal, repair and maintenance services (e.g. hairstyling) and governmental, quasi-governmental, and non-profit services. The medical examination suite in section 2.2 would fall in the domain of health care services while the flagship store examples discussed above can be positioned in the channel services category.

2.5 Concluding remarks

In this chapter we looked at the social, cultural and economical factors that shape the ambient intelligence landscape to better understand how ambient intelligence helps people in performing their everyday life activities and rituals. We learned how the ambient narrative interaction concept that was derived from dramaturgical and literary theory can represent dynamically changing intelligent environments in an generic, application independent way that facilitates a mass customization strategy towards ambient intelligence. In the next chapters we focus on ambient narratives in the retail domain and work our way towards a prototype system.

3

Eliciting Functional Retail Requirements

The concept of ambient narratives can represent a wide variety of intelligent environments in a modular way to implement mass customization strategies for ambient intelligence as we have seen in the previous chapter. The exposition of ambient narratives has deliberately been kept at an abstract, general level to prevent ourselves from taking refuge in a narrow set of applications and create artificial boundaries where there are none, as ambient intelligence also does not make this difference in its definition. There is no reason why technology should prevent a designer from authoring a smart hotel lobby environment where people could also lend or buy books. In order to validate this concept with real users (both consumers and producers of smart environments) and work our way from an abstract definition towards a concrete system design, we must choose for a particular application domain. In this chapter we refine the focus of this thesis to retail environments and intelligent shop windows in particular (section 3.3). Section 3.1 describes the insight generation process to understand the different stakeholders and their current way of working with regard to retail atmosphere creation. In section 3.2 we confront retail stakeholders with the ambient narrative concept and different ways to author ambient narratives. The analysis of the functional requirements for (intelligent shop window) ambient narratives is presented and summarized in sections 3.4 and 3.5.

3.1 Retail insight creation

To understand the retail domain better and in particular the role of intelligent environments in retail environments, we set up a number of interviews with retailers, retail and hospitality consultants and designers. In 13 sessions we spoke to 18 people with various backgrounds (retail management, hotel management, interior architecture, industrial design, cultural anthropology), levels of expertise (2-20 years) and positions (professional designers, store owners, independent consultants and retail store executives) in the Netherlands and Finland and asked them in a structured interview about their current way of working with regard to designing, configuring and installing interactive ambient effects, media and applications in retail. The contextual interview questions were adapted from [Mayhew, 1999]. This led to a number of insights for the design and implementation of smart retail environment detailed below.

Insight 1 ***Shopping is a multi-sensory experience for customers***

The first question we asked was what people thought of interactive immersive retail experiences in general. All participants underscored the importance of retail as a multisensory experience. Stores should be attractive and stimulate the senses. A few people commented that today the emphasis is on image and lighting but that scent, touch and audio are equally important to carry the experience. The smell of fresh coffee or baked bread, choice of materials and music may be more subtle but equally important. In general anything that can be done in a store to improve the feelings and emotions of customers in a positive way adds value. One retailer commented that he would lower the temperature in the store, change the carpet and display a fireplace movie on a TV screen to create a warm winter feeling when selling winter clothes.

Insight 2 ***Retailers can apply interactive technologies to create a personalized customer experience in both direct and indirect ways***

In terms of interactivity there were more differences. The majority of the participants were positive about the potential of interactivity but had several remarks. Both retailers and consultants commented that people do not want to be spied upon and that the effects should not be too distracting. When developing interactivity in the store, care should be taken that it is done from the customer's point of view, based on shopper insights instead of commonly accepted beliefs about shoppers. The personal approach is important especially for small stores. Instead of viewing personnel as a cost factor that needs to be eliminated, hiring good service personnel can lift the customer experience to a higher level. Rather than replacing the personal touch with interactive technology, interactive technology should assist shop employees to do their job well. One consultant remarked that interactive technology is not only useful for providing tailored information and setting the right atmosphere in the store but can also be applied to measure and detect events such as how often people touch which products for example. This data can then be logged and used in data-mining to improve the store layout and product offering. This shows there are many different ways to apply interactivity to enhance the customer experience. One retailer commented his employees already enter customers purchasing information and other sales remarks in a customer relationship management system. A shop employee can type in the name of a customer and get information about previous purchases, their likes and dislikes and so on to personalize the service to the customer. It would be useful if technology could recognize people that have already visited the store before so the service personnel could start a conversation with them he continued. Another participant argued that this personal approach can work well but needs to be sincere, authentic.

Insight 3 ***The retail experience needs to stay true to the identity of the store and its products***

The element of authenticity was mentioned several times in various discussions. Both the interactivity and multisensory aspect needs to stay true to the identity of the company and the products, services and experiences it offers, otherwise people will view the service offering as a cheap and fake one. This can also mean bringing people back in contact with the 'raw stuff' of what products and services are made of. Many (young) people have lost this contact and have little or no idea about what a farm is, how clothes are made or what it takes to create consumer electronic products. Some aspect of the experience lifecycle may be enhanced more than others without losing authenticity. A store can both entertain, educate and be a place where people buy products and services.

Insight 4 ***Shopping is a social activity for customers***

Several retailers and consultants added that shopping is also a social activity. Shopping malls and markets are places where people with similar needs and lifestyles meet each other and can form communities. This social aspect is often forgotten and could be supported with interactive technology instead of surpressed further using interactive information screens where people can find information.

Insight 5 ***Retail experience design is a multi-disciplinary effort***

The second question we posed was whether people had experience in designing, configuring, installing or programming lighting, media and ambient effects for shopping areas. The answers varied widely because of the different backgrounds, levels of expertise and positions participants had, which provided useful insights into the roles of the different stakeholders in the value chain as visualized in Figure 3.1. Note that depending on the complexity of the store design, one or more roles can be taken on by the retailer or other parties. The consultant group had no experience with designing, configuring, installing or programming lighting, media and ambient effects. Their work is at a more strategic level: In a dialogue with the retailer they define the business goals and the desired customer experience and translate this into a store concept that is then implemented by a design bureau. The designers' experience was mostly conceptual (storyboards, photos, sketches). Because experience design is multisensory, it is multi-disciplinary work that involves interior architects, product, lighting, sound, scent and graphics designers. The designers (one interior architect, two lighting designers, two industrial designers) in our group were similarly specialized. One of the designers commented that the construction of interactive retail environments was technically complex and too expensive. The lighting designers had experience with creating lighting plans and finetuning these plans on location together with installers. The retailers had experience with controlling the light levels, setting up a video projection, running screensavers on monitors and downloading images from the Internet. Much of the configuration, installation and programming of lighting, media and ambient effects is done manually or outsourced to specialized design bureaus.

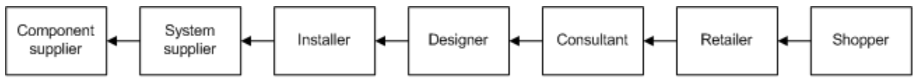


Figure 3.1: Retail stakeholders in store concept design and implementation

Insight 6 ***Keeping the smart retail environment up to date needs to become part of normal business for retail management***

The manager of a chain of flagship stores we interviewed reported to have experience with both lighting scene setting, LED lighting, audio/visual systems and building control systems to control heating and other environmental aspects. He commented one aspect that can easily be overlooked is shop employee training and updating the ambient effects in the store.

Although the time to switch between pictures, light settings and ambient effects takes less than a second, the time to update a store for a new collection can easily take two or more days of work said one retailer. Another retailer already spends a few hours a week to choose pictures and write texts for product presentation and placement in his store.

Insight 7 ***Ambient intelligence is to be taken into consideration from the concept phase onwards***

On the question if you design a new store concept when do you create the effects and media applications for this concept (concept, design or implementation phase), there was almost a uniform agreement that the ambient intelligence should be considered at the beginning of the process in the concept phase. Three of the participants said they would take technology into consideration in the design phase, after the store concept has been thought of.

Insight 8 ***Designers work both off-site and on-site and in an iterative way***

In terms of where people would design these effects and applications the group of small independent retailers preferred to work on location in an iterative manner whereas the consultants and designers were more inclined to work from their office or a combination of their office and location with feedback of their customer in this process. The retail chain reported they would design the audio, video and light effect playlists for a prototype store where they would test it all first before uploading the playlists and the content to a central repository from which all the other stores would look for new playlists and content to update their in-store experience.

Insight 9 ***Designers use a variety of separate design methods, techniques and tools in combination***

On the questions how do you program these effects and applications now and what tools and applications you typically use, the consultant and designer group employed a wide variety of different methods, techniques and tools. Sketches and storyboards made with pen and paper, digital photography (Photoshop), 3D visualization and walkthroughs (AutoCAD, 3D Studio Max), city trend tours and direct feedback on location were frequently mentioned. With regard to configuration and installation, professional lighting control systems and digital advertisement delivery systems for in-store marketing communication were being used by the larger retail chains. The more advanced interactive installations found in e.g. flagship stores are custom-built using commercial off-the-shelf hardware and software components. Although the content playlists and lighting may be (re)programmed (centrally) the interactive behavior is fixed. There is no easy way for interaction designers to change the mapping of sensors to actuators and how the system responds to shopper feedback and physical context changes (hardcoded into the application).

Insight 10 ***Retailers would like to change the appearance of their store more frequently in an easy and faster way***

The independent retailers we interviewed all wanted to change the appearance of their store on a more regular basis. At present this is done four times a year and sometimes only once. The consultants and designers mentioned this too. The flagship store participants commented that the lighting and video clips in their stores change during the day according to a playlist that is updated weekly. Two other retailers when asked how often they would reprogram the behavior and effects in his store said that the more flexibility, the more they would use it.

Insight 11 ***Designers discover and correct errors through experience***

When asked about which errors and problems people would typically encounter during configuration and programming and how they would discover and correct those issues participants commented they would use their senses to see, hear, feel and smell if something was not right. It should be noted that none of the participants had experience with scripting behavior for intelligent environments other than writing large but simple playlists and therefore may not have encountered the need for debugging tools.

On the question what are the main bottlenecks and problems in your current situation with regard to designing and programming smart retail environments lack of tools, high costs (and restricted budgets), available time (too much content, not enough time to manage) and lack of knowledge on how to apply it and develop quality content and scripts were mentioned. More flexibility and more tools to do the work faster and easier so that possibilities can be explored were seen as key areas of improvement. This reaction supports the argument for a mass customization approach and need for end-user programming environments for smart retail environments made in the research questions section (1.3.1).

3.2 Putting things in context

Can retail stakeholders understand the ambient narrative concept and which way of authoring ambient narratives do they prefer? To derive an answer for both questions we conducted a series of workshops with the same group of retailers, retail consultants and designers that were interviewed in the previous section.

3.2.1 User-generated ambient narratives

The concept of ambient narratives as spatial-temporal multi-sensory interactive narratives in mixed reality can be difficult to convey to people with a non-technical background. Instead of trying to explain this abstract concept directly, a more indirect approach was followed: First, we asked people to imagine their own store or a new store concept and draw this store layout on a large piece of paper. Next, participants were asked to write down a number of use-case scenarios from the point of view of the shopper. From the interviews we learned that many designers and consultants already work with experience maps and story boards. Not all participants were equally good at coming up with shopper scenarios in the one hour slot that was reserved for the workshop. If people had difficulty in imagining scenarios, we helped them by asking them what would happen if somebody would walk by the store, stood still in front of the shop window display, entered inside, browsed through the store, stand still in front of a product etc. After this step we presented them with the theatre service marketing framework as discussed in section 2.2 and asked them what their opinion was of this framework. Overall the participants liked this model and one of the consultants reported they use this idea of retail scripting already in practice. Two people indicated the importance of improvisation in service environments. Because an ambient narrative contains different intertwined storylines, the participants were asked to decompose the shopping scenarios into theatrical scenes, each having an action and situation description and write these scene descriptions on a post-it, which they had to position on the story layout. Although the process was structured, the participants had the freedom to fill in the different tasks as they saw fit because we wanted to see how they would approach these tasks and how they would for example write down these scene descriptions. Figures 3.2 and 3.3 show two examples we collected.

Fish outlet store

The idea behind the ‘fish outlet store’ ambient narrative in Figure 3.2 was to transform an ordinary store where you can buy fish and other seafood into an experience store where people besides buying seafood can be educated about our relationship with the sea and fishing, learn how to prepare fish. Rather than replacing the real thing with 3D simulations or videos, the participants argued that the immersive, interactive technology should render the authenticity of the products on sale. As people approach the store they see the catch of the day, the seafood that is recommended that day on large projection displays to seduce people to come inside (1). This is carried on inside the store (2). In the center of the store, aquariums with living fish and sea creatures give shoppers

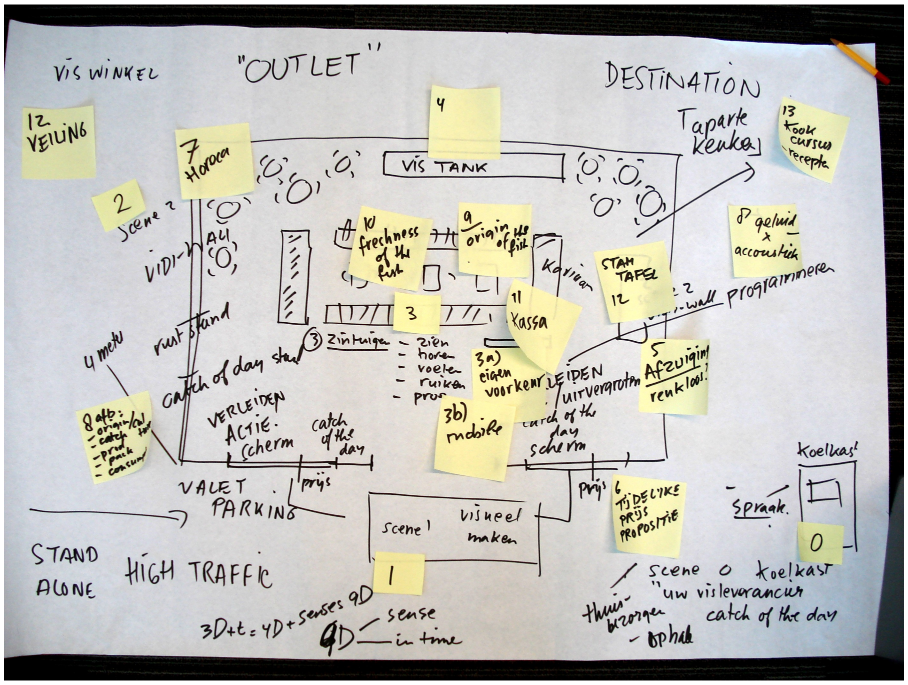


Figure 3.2: Fish outlet store ambient narrative

the feeling that the products sold here will be fresh (3,10). Recommendation of sea food is personalized to the customer (3a). In this area people can also experience where and how these fish live in the sea (9) using ambient media. In the top-left corner, there is a restaurant area where people can taste fish (7). In the top-right area there is a kitchen workshop where people can learn how to prepare seafood (13). This example illustrates how experience design integrates interior architecture, interactive media and ambient technologies. Ambient intelligence plays a role in the overall experience but does not play a dominant role, many of the post-its in this example do not even require ambient technologies per se.

Used clothing store concept

Another participant came up with a used clothing shop concept where people can buy secondhand objects such as clothes, books, radios, clocks and furniture, see Figure 3.3. Example scenes are (with actual text written in Dutch): As a person approaches the store, the age of the person is estimated and used to personalize the product presentation in the shop window through highlighting (*Leeftijd schatting; aanbeveling 80 product d.m.v. uitlichting*). If people

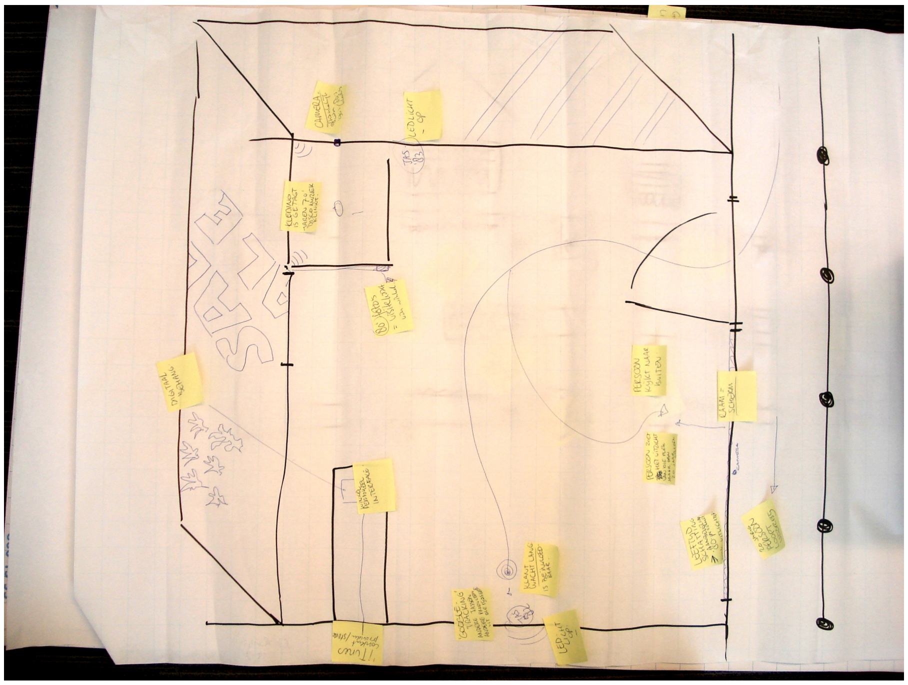


Figure 3.3: Used clothing ambient narrative

enter the shop they see a large digital canvas in the back of the store (*digitaal behang*) that can be controlled by store employees to react on the situation in the store (*winkel personeel interface*). When a shopper walks up to a jacket in the store and keeps standing in front of this object for a while, the jacket is highlighted along with other products and accessoires from the same style and period elsewhere in the store (*Google 'tracking', andere tassen, andere producten uit die tijdsgeest*). As a person brings a tagged piece of clothing of e.g. the seventies inside the dressing room, seventies music starts to play in the dressing room (*kleding is getagt, jaren 70 disco muziek klinkt*). The person can take a sepia photo of herself with the address of the store on the back.

Analysis

In total the workshops resulted in 12 different ambient narratives, 76 use-case scenarios and 117 post-its with scene descriptions. In addition to the fish outlet and used clothing store, participants came up with ambient narratives for an electronics store (2x), jewelry boutique, fashion shop, furniture store, design store, hairdresser salon, banking branch and a concierge service (physical place where people can design their own virtual alter ego and subscribe to different services). Concepts that were mentioned several times were product interaction (person picks up a product or otherwise interacts with a product in the store and e.g. the lighting changes), digital wallpaper, multi-sensory product and store presentations that react on external conditions (time of day, weather outside, people on the street), interactive shop window and in-store displays, personalized routing, co-creation environments (create your own jewelry, virtual avatar) and experience rooms that can be customized by shoppers.

The action descriptions of the collected scene post-its made use of visual information in the form of text, graphics and video on a variety of display devices, dynamic (colored) lighting, audio in the form of recorded speech, sound effects and background music, haptic feedback, fragrances and environmental effects (temperature, humidity levels). The actions varied from simple rendering to complex software applications. The situations described were analyzed on used context information parameters. A *context information parameter* can be defined as a single fact of information about an entity like the location of a person or the identity of an object. Context parameters found were the location of a person (actor) or device/object (prop) with regard to an area or object, the profile of a person or object, person orientation, the activity of a person or object (e.g. person standing, object switched off) and relation between a person and an object (e.g. person touches or looks at an object), date and time, light and temperature outside. See Table 3.1 for the list of context parameters found.

Next to the retail workshops an additional literature scenario survey was conducted to determine the kind of context information needed by ambient intelligent surroundings [Van de Heuvel, 2007]. 32 home and 32 retail scenarios were analyzed on used context parameters. In total 55 unique context parameters for the retail domain were found. The ten most frequently occurring parameters cover 59% of the retail domain scenarios in this study. These ten parameters were also found in the user-generated ambient narratives (Table 3.1). Adding actor profile and environmental condition brings the coverage to 69%. Context parameters that were found in the literature study but not in the workshops were more ‘exotic’ parameters such as human intentions and emotions, perceptual attributes of objects (e.g. color, shape), object-object and person-person relationships.

The style of writing on the post-its and the place of the post-its on the store layout also revealed useful information. In general, the consultants and some of the designers would describe scenes on an abstract conceptual level with words such as *veranderlijk, inspirerend, snel, zacht* (transient, inspiring, fast, soft) or first sketch the story in broad terms as in the fish outlet store case.

Context parameter
relative position of an actor/device/object to an absolute area
absolute position of an actor/device/object to an absolute area
relative position of an actor to a specific device/object
absolute position of an actor to a specific device/object
relation (touch, gaze) between an actor and a device/object
actor profile (name, role, age, gender)
activity of an actor (waiting, walking)
actor orientation
device profile (name, capabilities)
device state (on/off)
date and time interval
external light and temperature levels

Table 3.1: Context parameters found in user-generated retail ambient narratives

The other group of designers and the retailers in general were thinking more in concrete scenes as in the used clothing store case. Furthermore we learned that most participants focussed their attention on describing the action and creating the desired effect and much less on the situation that should trigger this action. One explanation may be because most people had little or no experience with programming intelligent behavior and were therefore not used to setting such rules. People would also use more post-its if the desired action and trigger for that action would not occur in the same area. For example, a digital wallpaper post-it would be placed on a back wall and the user interface (trigger) behind the counter.

3.2.2 End-user programming strategies

In order to figure out which way of authoring ambient narratives designers and retailers would prefer we proposed four different programming strategies to the participants of the workshop sessions afterwards and asked them what they thought about these individual strategies. To make sure everybody understood it was not about the scenario but about the way the scenario was programmed, we stated the same scene to start from in each of the four cases and explicitly mentioned this. The scene we picked is one in which a transparent display on a shop window shows an interactive shop catalogue when a person is standing close in front of it.

Figure 3.4 shows a scenario where the retail designer sits in front of his PC and looks at a 3D simulation of her store. The designer sees the 3D environment and walks with her virtual character to the shop window and sees

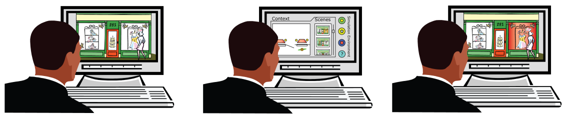


Figure 3.4: Programming in a 3D simulation environment

that nothing happens (left). The designer switches to her programming environment and is presented with an overview of the location of people, devices and objects in his store and which scenes are currently active (middle). The designer presses a button to create a new scene for the shop window. First she draws the stage in front of the shop window, then she associates a customer actor and shop window device prop to this stage. She then sets the shop window to 'interaction' and saves the newly created scene. The designer switches back to the 3D simulation and sees a picture of the interactive shop catalogue on the virtual shop window (right). This programming strategy is based on the idea of using a 3D simulation for rapid prototyping of ambient intelligence using a game engine as put forward by e.g. [Barton & Vijayaraghavan, 2003; Shirehjini & Klar, 2005]. This scenario also partially supports Insight 8 (*Retail designers work both off-site and on-site and in an iterative way*) as put forward in section 3.1.

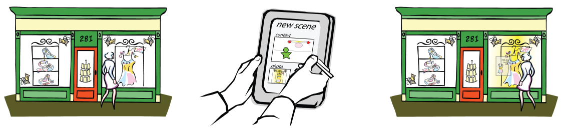


Figure 3.5: Programming on location using a portable device

Figure 3.5 presents a scenario where the retail designer is physically walking through the store and experiences the environment. The designer is detected by the first shop window, but nothing happens (left). The designer picks up the portable device and is presented with a real-time overview of the situation in the store and which scenes are currently active. As in the 3D scenario, the designer sets the context situation and device actions using the graphical user interface and saves the newly created scene (middle). The designer then walks up to the first shop window and sees that the transparent display in the shop window switches to the interactive shop window catalogue (right). This programming strategy is an example of visual programming on location using a hand-held device for context-aware applications as described by e.g. [Humble, Crabtree & Hemmings, 2003; Li, Hong & Landay, 2004; Weal & Hornecker, 2006]. This approach is backed up by Insight 11 (*Errors are discovered and corrected through experience*).

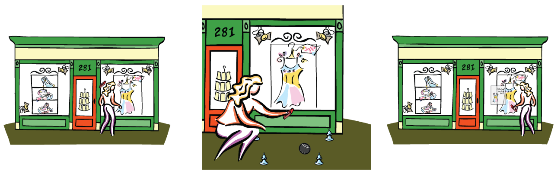


Figure 3.6: Programming by demonstration using physical objects

Instead of letting designers specify context-aware rules explicitly, programming by demonstration, also refered to as programming by example [Cypher, 1993; Lieberman, 2001], infers these rules from the behavior of users. This method is proposed by the third scenario in combination with tangible objects that represent people, devices and corners of sensitive areas: Figure 3.6 shows a scenario where the retail designer is physically walking through the store and experiences the environment as in the previous scenario (left), but instead of taking a portable device, the designer walks up to a tabletop interface and sees an overview of the location of people, devices and objects in the store and which scenes are currently active. To create a new scene, the designer demonstrates the desired context situation to the system by physically placing tagged objects that represent actors, props and stage corners (middle). When the designer is done, she can save the scene and remove the tagged objects. The designer walks up to the first shop window and sees that the transparent display in the shop window switches to the interactive shop window catalogue (right). Examples of programming by demonstration systems for context-aware applications are described by [Dey, Hamid & Beckmann, 2004; Hartmann, Abdulla & Mittal, 2007]. A survey of physical language design can be found in [McNerney, 2004].

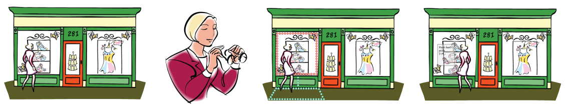


Figure 3.7: Programming in augmented reality

Figure 3.7 shows a scenario where the retail designer is physically walking through the store and experiences the environment as in the previous two scenarios. Instead of using tangible objects or a portable screen, the designer puts on an augmented reality interface and sees through these glasses which scenes are currently active and deactive and real-time information about people, devices and objects. To create a new scene, the designer makes gestures to draw a zone in front of the first shop window and selects from a menu actors and props to associate to the newly defined stage. The designer then takes off the augmented reality interface and walks up to the shop window to see that it presents the interactive shop window catalogue. [Sandor, Bell & Olwal, 2004] describe a demo where end-users can configure the setup of a distributed user interface based on a shared augmented reality. Augmented reality has also been proposed for visualizing architecture, see e.g. [Thomas, Piekarski & Gunther, 1999; Krogh, 2000].

Results

Participants ranked the 3D simulation scenario highest in terms of usability (mean = 5.8, median = 6, $\sigma = 1.6$ on a 7-point scale) closely followed by the on location using portable device scenario (mean = 5.5, median = 6, $\sigma = 1.4$). The programming by demonstration scenario was least appreciated (mean = 3.7, median = 4, $\sigma = 1.5$), the augmented reality scenario ranked third (mean = 4.5, median = 5, $\sigma = 2.7$).

The 3D simulation environment was seen as a useful tool by both designers and to a lesser extent also retailers. Some participants commented they would already use 3D walkthroughs to visualize store concepts so this would fit with their practice. Others commented that the testing and simulation of especially lighting is difficult and that therefore a 3D simulation alone would not be sufficient. Both designers and retailers liked the fact that this method would enable them to program scenes at home or in the office. The programming on location using a portable device was primarily seen as a tuning tool for the retailer and the installer. The portable device was not seen as a problem, provided the user interface would be easy to use.

Participants had more doubts with the other scenarios. Most participants could not imagine themselves using the programming by demonstration using tangible objects scenario. Two designers liked the one to one mapping with the real world and thought this scenario would be helpful for novice users but the other participants thought this scenario was inefficient and time consuming. User evaluations of in-situ authoring tools that implement a programming by demonstration strategy show that users both liked these systems and are able to successfully create applications with these tools [Dey, Hamid & Beckmann, 2004; Hartmann, Abdulla & Mittal, 2007]. Programming by demonstration can also increase effiency [Hartmann, Abdulla & Mittal, 2007]. Research done in tangible programming languages in the domestic domain [Blackwell

& Hague, 2001; McNerney, 2004] and for children [Montemayor, Druin & Chipman, 2004; Horn & Jacob, 2006] indicates that end-users like this way of programming behavior. Our hypothesis is that tangible programming is less interesting for retail and professional application domains than for home and leisure environments where efficiency is less of an issue and that it is primarily the tangible aspect in the programming by demonstration using tangible object scenario that retailers, consultants and designers do not like. The reactions on the augmented reality scenario were more mixed. Some participants thought this visualization was the most intuitive and interesting one to use, others thought this was too complicated and long-term. This latter opinion is subscribed by e.g. [Azuma, 1999; Navab, 2003] who discuss a number of technological challenges that the augmented reality research community needs to face in order to succeed in real life situations, including better position tracking and dealing with unstructured dynamically changing environments.

On the question if people would use some scenarios in combination and if so in what order, a few participants commented they would use the 3D simulation to design the intelligent behavior of the store at home or in the office and the portable device or augmented reality glasses to finetune effects and lighting levels on location. This result confirms our earlier findings that retail experience design is a multi-disciplinary effort (Insight 5) which involves a variety of separate design methods, techniques and tools in combination (Insight 9). Similar results are reported by [Weal & Hornecker, 2006] who discuss requirements for in-situ authoring of location-based experiences based on the lessons learned with an authoring environment to annotate a historic country estate with location-based media. What they found is that the incremental building up and changing of content is an integral part of the curator's work practice and that this is supported by in-situ authoring, however they also discovered there may be a need for an explicit effort to fill the system with basic content and that there should be different ways to define and change content both on location and afterwards.

After their feedback on the four proposed authoring strategies a number of other questions were asked. On the question "do you think it is relevant for a store owner or employee to adapt the retail experience" all participants agreed positively. On the same question but now for the customer instead of the retailer, only a few participants answered that they saw a need (e.g. to personalize the fitting room). Most participants wanted to have control over how the store appears. For suppliers answers were mixed. Some people thought it could work for shop-in-shop situations where a supplier has its own product booth that he can customize. One designer and retailer commented that suppliers could deliver audiovisual content along with their products that could be used by the retail designer in the retail experience. Another group argued that the retailer should be in complete control. The answers on the question do you think it is helpful to share programmed scenes with others and modify examples made by others were also mixed. Some participants argued that this would help them to learn from others, others said they would be careful because they did not want their smart retal environment copied elsewhere. It should be seen as copyrighted content said one designer. The question do you think it is useful to centrally change the behavior of individual stores was answered with a strong yes by all participants except the independent retailers. However it should still be possible to adjust this behavior locally noted a number of people.

3.3 Intelligent shop window ambient narratives

One type of scenario that was mentioned often during the retail workshops was that of a shop window that reacts to the presence and activity of people. An example already discussed in section 3.2.1 is that of a shop window that highlight products in the shop window based upon the profile of the person standing in front of the shop window display. Another example found during the workshops was a transparent display in the shop window that tries to attract people walking by with moving images to invite them to come inside. One designer suggested to place the shop window in the back of a hairdressing salon concept to create the impression of an open inviting space. Several retailers and consultants also commented that the shop window is crucial in making the first right impression and draw people inside the store. Intelligent shop window environments therefore seem a rich and interesting area to look into further.

With these functions of shop windows and example scenarios in mind a intelligent shop window prototype was built in a user-centered way [Van Loenen, Lashina & Van Doorn, 2006]. The goal of this prototype was to demonstrate the concept to retailers and shoppers and later compare this custom-built, monolithic version with a modular ambient narrative version. This exercise is useful to determine the steps needed to refactor a traditional user-centered designed application into the modular ambient narrative model and the problems that are encountered during this process. Furthermore, it allows evaluation of both approaches in terms of system performance and functionality. The shopper in the end does not care about whether or not the retail experience is based on a modular platform as long as the functionality and performance is the same. The results of this system evaluation can be found in section 7.2. In the remainder of this section the use case scenario of the first intelligent shop window prototype is discussed. This scenario integrates different elements that came up during the retail workshops and that were found in related work on public ambient displays and places them in a fashion store setting.

3.3.1 Use case scenario

Emily and Olivia have an afternoon off from work and decide to go shopping for fun. When they approach the high-end fashion store near the central plaza of the shopping mall, they see moving images of the latest women collection on the shop window. Attracted by the sight of pictures that seem to be hanging in mid air, Emily moves closer to the shop. The shop window tries to capture her attention by playing music and showing pictures near her. Emily follows the music and turns towards the shop window. Soon her attention falls on the pair of shoes on display. After a few moments she sees information about the shoes appear on the shop window in front of her. The shoes in the store are also highlighted. Come over here*, she says to Olivia. As Olivia approaches the shop window, she is not greeted with music and imagery because Emily is still interacting with the shop window. Olivia walks up to the shop window section next to Emily and points to a dress behind the shop window. Immediately, a product catalog appears on the shop window and Olivia starts to browse through the collection and sees more items she likes to buy. Olivia and Emily decide to go inside.* [Van Loenen, Lashina & Van Doorn, 2006]

Several different elements can be identified in this scenario. First, the interaction style depends on the distance of the shopper to the shop window. When people are at a distance, the shop window tries to attract people with images as they move closer the product slideshow disappears and an interative shop catalogue appears that allows shoppers to browse the collection. This idea is adapted from [Vogel & Balakrishnan, 2004] who identity four interaction phases (ambient display, implicit, subtle and personal interaction) facilitating transitions from implicit to explicit, public to personal, interaction.

Unlike most public ambient displays, the intelligent shop window does not replace but augments the view of the products on display behind the shop window to allow shoppers to see inside and make the store look larger. To achieve this effect a shop window can equipped with a holographic foil that enables a beamer under a specific angle to project images and videos on the glass. This technique has been used in a couple of Ralph Lauren stores [Toler, 2007]. Besides conveying information, the displays simulatenously try to attract people with aesthetically pleasing graphics. This is combination is also proposed by e.g. [Fogarty, Forlizzi & Hudson, 2001] who describe decorative public ambient displays that contain information.

Directional audio and an interactive shop catalogue screen that is smaller than the shop window itself provides a form of narrowcasting that protects people's privacy. In the scenario two shoppers can browse through the shop collection at the same time. A public ambient display framework that also supports multiple users simulateneously accessing information that contains both public and personal elements is presented by [Cao, Olivier & Jackson, 2008]. An example of a public ambient display in a hospital setting with both public and private information is discussed in [O'Neill, Woodgate & Kostakos, 2004].

Besides implicit forms of interaction such as the position of a person in front of the shop window, the scenario also illustrates different ways of explicit interaction: A shopper can bring up information about a product on a transparent display in the shop window by either looking at the product or touching the glass. Most examples of public ambient displays are based on touch see e.g. [Vogel & Balakrishnan, 2004; O'Neill, Woodgate & Kostakos, 2004; Toler, 2007], a multi-touch example is presented in [Peltonen, Kurvinen, Salovaara & Jacucci, 2008] who report positive results with this form of explicit interaction in the context of a photo sharing application in the city of Helsinki. Gaze feedback as a modality for interacting with ambient intelligence environments is explored by [Gephner, Simonin & Carbonell, 2007]. [Qvarfordt & Zhai, 2005] couples gaze feedback to information presentation for a desktop tourist application. When a person looks at a map on a screen, an arousal score is calculated for each object in the map. If this score reaches a certain predefined threshold, the object is activated and the information about this object is presented on the screen. [Prasov & Chai, 2006] describes how a conversational interface can use gaze direction to determine where the attention of the user is to eliminate ambiguity in speech recognition. An overview of different interaction technologies for large displays is given by [Bierz, 2006].

Finally, the scenario shows how feedback is given using multiple modalities simultaneously. When people interact with the shop window they will not only receive feedback through the screen, but also via light and audio cues. This functionality is interesting to visually impaired people.

Because many retailers expressed their concern about letting customers change the appearance of their store front, the scenario does not support shoppers to e.g. annotate the shop window display with electronic notes or other media content as discussed by [Peltonen, Salovaara, Jacucci & Ilmonen, 2007; Tang, Finke & Blackstock, 2008].

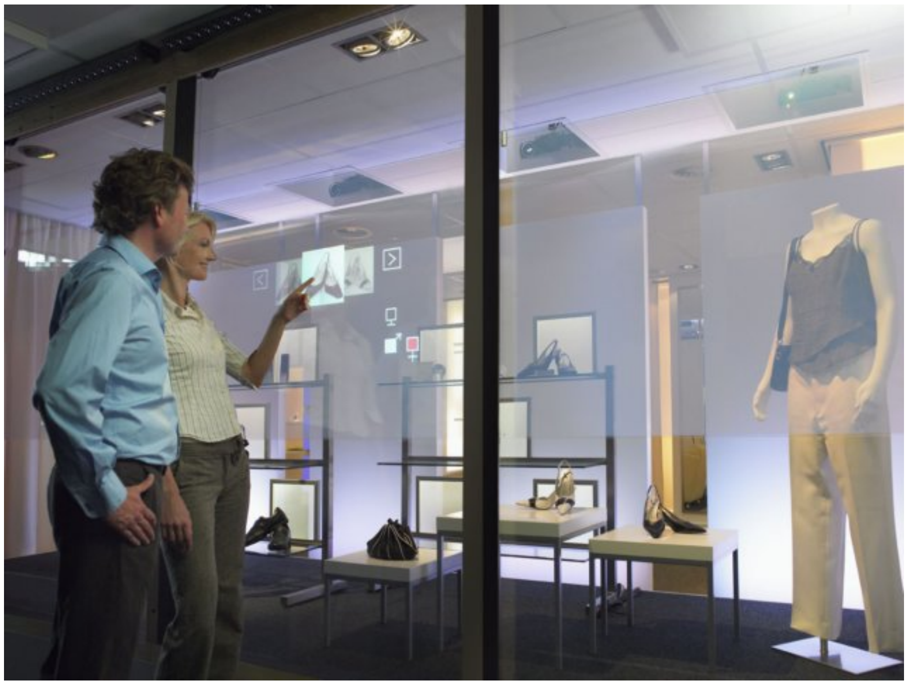


Figure 3.8: Intelligent shop window set-up in ShopLab

Figure 3.8 shows the first prototype developed and integrated in ShopLab, a controlled but realistic shop environment on the High Tech Campus in Eindhoven for usability and feasibility testing. The prototype has been shown to a large number of both internal and external visitors since the summer of 2006 and generated a lot of interest from both retailers and potential shoppers. The possibility to attract people, show product information (also when the shop is closed) is liked by retailers. Shoppers responded that they would not feel strange using the intelligent shop window in a public environment and said they would be interested in obtaining more information about products on display via the intelligent shop window. A user study on the different interaction styles (touch and gaze interaction) and feedback mechanisms (visual, lighting and mechanical turn tables) is described by [Kessels, 2006].

3.4 Requirements analysis

The derived ambient narrative concept, collected user insights (3.1), user-generated retail ambient narratives (3.2.1), results of the user study on end-user programming strategies (3.2.2) and the intelligent shop window user scenario (3.3.1) can be combined and analyzed for requirements to be placed on an ambient narrative system with authoring support for smart retail environments.

To structure this requirement analysis process we first considered the basic characteristics of the ambient narrative concept. This led to a first set of functional requirements and created a mental box in which we would have to position the other functional requirements. This decision is justified by the user-generated ambient narratives that were collected and the feedback we received from participants. The next step was to look into the user-generated retail ambient narratives and the shop window scenario and search for application and system requirements with the performance and functionality of the system in normal operation in mind. The formal model in the next chapter is derived from these two sets of functional requirements. Third, we shifted our attention to supporting end-users to control the lifecycle of ambient narratives. The collected user insights and the results of the user study on different authoring strategies were used as input. This set of functional requirements was used to derive the overall system architecture that will be discussed in Chapter 6.

The outcome of this elicitation process is a list of thirty requirements that we have grouped into four categories: *ambient narrative concept implementation*, *run-time system performance*, *end-user authoring user interface* and *end-user software engineering support*.

The *ambient narrative concept* requirements cover the aspects a system needs to address to implement the high-level ambient narrative interaction model. The *run-time system behavior* requirements deal with the functionality and performance of the system in normal use. The *end-user authoring support* requirements include aspects such as visualization of the internal and external state of the environment and control of designers over the behavior of the system. The *end-user software engineering support* requirements deal with controlling the lifecycle of ambient narrative fragments by end-users such as testing, porting and versioning of ambient narratives.

The next sections contain the list of requirements for each category that an ambient narrative platform, which supports retailers and designers to create, deploy and maintain their own smart retail environments should fulfill. When certain requirements support or enable other requirements this will be explicitly mentioned. Ease of use of the authoring environment, creative freedom to design the behavior of the intelligent environment in the way designers imagine, and total cost of ownership of the total system are key factors that are addressed by many of these requirements and therefore not mentioned individually.

3.4.1 Ambient narrative concept requirements

Ambient narrative concept implementation requirements deal with the system aspects for implementing the ambient narrative interaction model.

Requirement AN.1. ***Social scripts enhanced with ambient technology***

The ambient narrative interaction model integrates ambient media and applications into the dramaturgical theory of social life as discussed in sections 2.1 and 2.2. People perform culturally defined social scripts that can be detected and enhanced with ambient technology. When a social script is being performed and recognized, the associated device action(s) must be rendered on the specified device(s).

Requirement AN.2. ***Mass customization of ambient narrative fragments***

To support mass customization strategy for ambient intelligence the intelligent environment needs to be broken down into smaller fragments (Requirement A.1) that are assembled based on customer specifications in a customized product (ambient intelligence). The customer specifications are elicited through the collective performance of actors in the environment: By interacting with the ambient narrative and performing social scripts, actors implicitly select and sequence the ambient intelligence fragments into a story. Ambient intelligence is that part of the emerging story that is conveyed through the devices that surround the actors in the environment and perceived by users (section 2.3.3).

Requirement AN.3. ***Separation of ambience and intelligence by a network***

The application intelligence should be disembodied from the tangible, physical devices that render the ambience as much as possible so we can view ambient intelligence as an information product or service that is delivered to (the devices surrounding) the user. This has a number of advantages. First, the rendering devices can be kept simple, standardized and mass produced at low cost. They can be installed once and do not have to be replaced. Second, information products have the advantage over physical products that they carry no weight, do not occupy shelf space and can be produced and delivered almost instantly. This makes mass customization of intangible information products often economically more attractive than mass customization of physical products, especially in the context of a global network society that operates in a demand-driven way [Van Doorn & De Vries, 2006].

Requirement AN.4. ***Run-time modification of ambient narrative fragments***

To support experience co-creation environments and end-user programming of ambient narratives (section 2.3.3, human actors and/or software agents working on behalf of human actors should be able to add new fragments, edit existing fragments or delete old fragments while the system is running. In an (end-user) authoring tool for ambient narratives this modification is done in an explicit way: Authors use the authoring environment to modify the ambient narrative. Fragments may also be *generated* by other fragments. In this case the set of fragments in the ambient narrative changes indirectly as the result of the collective performance of actors in the environment.

Run-time modification of ambient narrative fragments is also interesting if we consider a shop as a place where people with similar interests meet each other (Insight 4). Customers could personalize the ambient intelligence in the shop to their interests.

3.4.2 Run-time system behavior requirements

Run-time system behavior requirements cover the functionality and performance of the system that are not specific to the ambient narrative concept and do not address authoring support. The run-time system behavior requirements can further be divided into functionality requirements (Requirements RT.1 to RT.4), performance requirements (Requirements RT.5 to RT.7) and flexibility requirements (Requirements RT.8 and RT.9).

Requirement RT.1 ***React on implicit context changes***

The list of context parameters found in the user-generated retail ambient narratives can be found in Table 3.1 (p.41). In the intelligent shop window scenario in section 3.3 the following context parameters can be found: relative position of a person and device to an area (e.g. person standing in front of a shop window), device identifier (e.g. one of the four transparent displays) and the relation between a person and an object (e.g. person looking at a product in the shop window). The system should keep a model of these context parameters and react to changes of these parameters.

Requirement RT.2 ***React on explicit user feedback***

Next to implicit context changes the system should also respond to different forms of more explicit user feedback coming from applications that were started on devices. Clicking with a mouse on a button in a screen-based interactive media presentation in the store or pointing by hand to an item in the interactive catalogue application on the transparent shop window display in the shop window scenario can result in the activation or deactivation of other ambient intelligence fragments (e.g. highlight a tile behind the product on the shelf).

Requirement RT.3 ***React on state changes***

The system should not only react to external changes (implicit context changes or explicit feedback) but also to changes in its internal narrative state. This requirement was not found during the workshops but can be derived from the intelligent shop window scenario. If there is a person standing in front of one of the shop windows and nobody in the area further away from the shop windows, both the product presentation slideshow and the interactive shop catalogue will be started whereas in the scenario only the interactive shop catalogue should be active. System session state information is needed to support this behavior.

Requirement RT.4 ***Timing and synchronization of actions over multiple devices***

From the interviews we learned that shopping is a multi-sensory experience (Insight 1) and that interactive technologies can add to a more personalized customer experience in different ways (Insight 2). Therefore we could expect a wide variety of different devices and applications running on those devices. This was confirmed by the workshops (section 3.2.1). The system should therefore be able to control which applications, media and ambient effects on those devices are started and at what absolute or relative time after a social script with its associated device actions has become active.

Requirement RT.5 ***Users do not experience a difference in performance with a custom-built system***

Customers in general will not be interested in whether the smart retail environment implements the ambient narrative interaction model or is custom-built as long as the system works smoothly. The overall system should have a fast response time and work without many glitches in normal operation. There must be no noticeable difference in run-time performance with a custom-built system that implements the same use case scenario(s).

Requirement RT.6 ***Scalable in terms of sensors and actuators***

In a realistic smart retail environment there will be many sensors that generate streams of data that need to be processed sufficiently fast in order to make timely decisions about which actuators should be controlled. The more sensor input available, the more context parameters can also be sensed and the more situations can be detected by the intelligent environment and responded upon.

The expressiveness of the system is bounded by how much sensor data it can process in real-time (Requirement RT.5). The run-time performance of the system should therefore be able to address a number of sensors some of them generating large amounts of data. Filtering sensor data is one way to reduce this sensor data overload but the system should ensure no relevant information is discarded during this process.

On the actuator side, retail designers of smart retail environments should be able to add more devices such as extra transparent displays as in the intelligent shop window scenario without experiencing a degradation in run-time performance.

The intelligent shop window user scenario only covers a small area of the shop and does not need to control many sensors and actuators.

Requirement RT.7 ***Robust against sensor and actuator failures***

Retail environments need to be open all day during the week and often also in the weekends. The shop (window) cannot be closed to perform maintenance work during the day as this will cause lost revenues for the retailer. The system should be robust against (temporary) sensor and actuator failures. If one sensor or actuator fails the system should continue to work as best as possible by graceful degradation in terms of functionality offered.

For a research prototype this requirement can be relaxed.

Requirement RT.8 ***Plug and play actuators***

Among the ambient intelligence fragments collected, a number of fragments were identified that involved a portable device as part of the retail experience. To support portable devices that can play a role in smart retail environments, for example a mobile phone that will show information about a product on display if it is close to that product, the system should automatically recognize new devices and know when devices are no longer present. In the intelligent shop window scenario this functionality is not necessarily needed.

Another reason for supporting *plug and play* devices is that this allows the system to grow with the retailer's business. A retailer could start small with a single transparent display behind a shop window for example and buy extra transparent displays and LED wall washers to illuminate the back wall later. This reason alone would be insufficient for this requirement as the installation of new devices (e.g. cabling, mounting equipment) probably requires the store to be (partially) closed and the system taken offline anyhow.

Requirement RT.9 ***Plug and play sensors***

Similarly the system could support plug and play sensors. This requirement is considered to be less important than the previous one as none of the user-generated retail ambient narratives contained a fragment that used this feature. However, it is possible to imagine a scenario where a person comes into a room wearing a body sensor that detects biometric signals that will cause an ambient intelligence fragment to be activated that sets the light in the room to reflect the mood of the person.

3.4.3 Authoring user interface requirements

The authoring support requirements deal with visualization of the state of the ambient narrative system so designers can understand how the system works (Requirements UI.1 to UI.5) and the behavior of the intelligent environment that should be placed under control of the designer (Requirements UI.6 to UI.8)

Requirement UI.1 ***Summary of active and inactive ambient intelligence fragments***

To change the behavior of the ambient narrative, users should know which ambient intelligence fragments are already active and which fragments are stored in the system but currently not activated. The authoring tool should visualize this information in an intuitive way so that designers can quickly figure out what the state of the system is.

This overview also helps the retailer or retail management of a chain of stores to see which scripts are being performed by customers and service personnel in a shop at the moment. The activated scripts could be logged together with their timestamp to collect statistics about the activities that take place in a store and gather insights about shoppers. An example of tracking shopping behavior, demographics and trip types is provided by [VideoMining, 2008].

Requirement UI.2 ***Summary of current context situation***

The current model of the context as sensed by the system should be visualized so the designer or system operator can check if people, devices and objects are recognized by the system. This helps the designer to quickly figure out if all devices are turned on and work correctly and also assists the designer in determining why a particular fragment is currently not activated.

Requirement UI.3 ***Identification of ambient intelligence fragments***

The designer should be able to quickly identify which media-enhanced social script is presented by which ambient intelligence fragment in the list of active and inactive fragments (Requirement E.1). Most photo browsing tools use image thumbnails so that designers can quickly browse relatively small image databases [Kang, Bederson & Suh, 2007]. Similarly, designers could take a photo or make a video clip of the effect of the ambient intelligence fragment to add an illustration along with the name of a new fragment to find a particular fragment back more easily.

Requirement UI.4 ***Overview of ambient intelligence fragment situation***

The social script that triggers the device actions of an ambient intelligence fragment needs to be visualized to the designer in an intuitive way if the designer views or edits the fragment in the authoring tool. As we adopt a dramaturgical framework in a retail setting, the social script is expressed in restrictions on stage (physical location of the service), performance (service offering), actors (service personnel), audience (customers), props (devices and objects involved) and script (business process). This terminology should also be used in the design of the user interface to create a clear connection between the technical authoring environment and a service marketing strategy that adopts the same underlying dramaturgical theory so that it becomes a tool that is conceptually integrated in a broader service management methodology.

Requirement UI.5 ***Overview of ambient intelligence fragment action***

The timing and synchronization of device actions (Requirement R.4) described in the ambient intelligence fragment should also be visualized in a way that is easy to comprehend by end-users. Designers should be able to navigate in this action timeline and preview the effects of these actions to deal with complex timed media presentations with ambient effects. Commercial DVD and Web authoring tools such as [Apple, 2008; Adobe, 2008] use a similar way of visualizing timed media clips and actions to authors.

Requirement UI.6 ***Control over the activation of ambient intelligence fragments***

The authoring tool should allow designers to control the behavior of the ambient narrative on a number of levels (Requirement UI.6 to UI.8). First, the designer should be able to set and edit restrictions on a social script. This allows the designer to control the activation of an ambient intelligence fragment.

Requirement UI.7 ***Control over the timing and synchronization of actions in ambient intelligence fragments***

If an ambient intelligence fragment is activated it should result in an effect on the environment. To associate a media item, ambient effect or application to a social script, the designer should be able to set and edit the timing and synchronization of device actions.

Requirement UI.8 ***Control over the action content in ambient intelligence fragments***

Designers not only want to choose between predefined actions but also create new actions. In the intelligent shop window scenario designer not only want to choose between an attraction or interaction mode of an intelligent shop window application that runs on a device but also add new actions or change the content. The product presentation slideshow for example needs to be updated because certain products are no longer available or the style and presentation must be changed to better reflect the store's brand image. The authoring environment should therefore provide support to control the content of actions in ambient intelligence fragments.

3.4.4 End-user software engineering support requirements

The end-user software engineering requirements deal with controlling the lifecycle of ambient narrative fragments by end-users such as testing, modifying and versioning of ambient narratives (Requirements SE.1 to SE.5). Related this support is the flexibility to program device actions and use new sensors directly in the end-user programming environment (Requirements SE.6 and SE.7).

Requirement SE.1 ***Simulation of the behavior of ambient narratives***

The user feedback on the end-user programming strategies discussed in section 3.2.2 showed designers and retailers liked the idea of having a 3D simulation of their store or a store concept they would be working on, both as a design tool and as a communication tool. The authoring environment should therefore support designers to test and present the behavior of their self-made ambient narratives in a 3D simulation tool.

Requirement SE.2 ***Portability of ambient narratives***

Because retail designers work both off-site and on-site and in an iterative way (Insight 8) and use a variety of separate design methods, techniques and tools (Insight 9) ambient narratives designed and tested in the off-site 3D simulation should work on the on-site live system and vice-versa. This was confirmed by the user feedback on the end-user programming strategies. Participants mentioned they would use different strategies in combination (section 3.2.2). This requires that the ambient intelligence fragments can easily be ported from one authoring system to another.

Requirement SE.3 ***Versioning of ambient narratives***

With end-user authoring support in place users can change the appearance of their store more frequently in an easy and faster way (Insight 10). If designers start to create new ambient intelligence fragments and ambient narratives a possible versioning and maintenance problem can be created. A structured process needs to be installed to ensure that someone is responsible for keeping the smart retail environment up-to-date (Insight 6). The authoring environment can support this process by giving designers the possibility to save ambient narratives and give them a name or version number. This is particularly relevant to larger retail chains that want to be able to centrally control the behavior of individual shops. Ambient narratives can be uploaded to a central Intranet site where they can be downloaded and used by the individual stores. This requirement depends on SE.3.

Requirement SE.4 ***Modification of ambient narratives on location***

Besides the 3D simulation environment for testing the behavior of an ambient narrative and communicating it to customers, the user feedback on the end-user programming strategies discussed in section 3.2.2 also revealed that retail designers would like to be able to modify an ambient narrative on location using a portable device to finetune ambient effects such as light levels in their shop (Insight 10) and be able to change the behavior of their store more frequently (Insight 11).

Requirement SE.5 ***Debugging the behavior of ambient narratives***

Professional software engineers spend much time debugging programs. Integrated development environments (IDE) for managing software such as [Microsoft, 2008; Eclipse, 2008] provide their users the possibility to e.g. step through the source code and inspect variables to quickly find and correct errors. Similarly, an end-user authoring tool for smart retail environments should offer designers support to detect and correct errors. Debugging support is also provided through the visualization of the ambient narrative (Requirements UI.1 and UI.2) as this visualization helps to understand the current state of the system and reason why a particular fragment is not active. Because we learned that retail designers discover and correct errors through experience (Insight 11), Requirement SE.4 can also be seen as a means of providing debugging support.

The more functionality an authoring environment offers under control of designers, the more complex user-made applications may become and the larger the need for debugging support. This requirement therefore depends on the expressiveness of the language model (AN.1 to AN.4 and RT.1 to RT.4) and in particular which of this functionality is exposed to end-users in the authoring tool (UI.6 to UI.8). Because the user-generated ambient narratives we collected had little or no dependencies between fragments specified by the participants and used relatively simple if-then rules, this requirement has a lower priority than the other end-user authoring tool requirements.

Requirement SE.6 ***Support plug and write actuators***

If a new actuator is connected to the system, it should not only be automatically recognized by the system and used in ambient intelligence fragments that define actions for this type of actuator (Requirement RT.8) but also appear in the authoring environment: Designers should be able to define new or modified ambient intelligence fragments that use both these and already present actuators in the smart retail environment to set context-aware actions.

Unlike the plug and play support for actuators that is needed to support scenarios with portable devices that enter and leave an ambient narrative, the authoring tool could be restarted after a new device is added. Run-time *plug and write* support is therefore not necessarily needed.

Requirement SE.7 ***Support plug and write sensors***

Similarly, if a new sensor is detected by the system, designers should be able to define new or modified ambient intelligence fragments that take advantage of this increased sensor coverage or functionality. Run-time plug and write support in the authoring tool would be convenient but not necessary as in case of Requirement SE.6.

3.5 Overview of requirements

Table 3.2 presents and prioritizes the functional requirements that will be translated into formal system requirements which can be placed on a system architecture. Requirements with a priority score of 1 must be fulfilled, requirements with a lower score influence the quality of the final design but do not invalidate this design in the author's opinion (for argumentation we refer to the specific requirement text above).

Requirement		Priority
AN.1	Social scripts enhanced with ambient technology	1
AN.2	Mass customization of ambient narrative fragments	1
AN.3	Separation of ambience and intelligence by a network	1
AN.4	Run-time modification of ambient narrative fragments	1
RT.1	React on implicit context changes	1
RT.2	React on explicit user feedback	1
RT.3	React on state changes	1
RT.4	Timing and synchronization of actions over multiple devices	1
RT.5	No noticeable difference in performance with a custom-built system	1
RT.6	Scalable in terms of sensors and actuators	2
RT.7	Robust against sensor and actuator failures	2
RT.8	Plug and play actuators	3
RT.9	Plug and play sensors	3
UI.1	View summary of active and inactive ambient intelligence fragments	1
UI.2	View summary of current context situation	1
UI.3	Identication of ambient intelligence fragments	1
UI.4	Overview of ambient intelligence fragment situation	1
UI.5	Overview of ambient intelligence fragment action	1
UI.6	Control over the activation of ambient intelligence fragments	1
UI.7	Control over the timing and synchronization of actions in fragments	1
UI.8	Control over the action content in ambient intelligence fragments	1
SE.1	Simulation of the behavior of ambient narratives	1
SE.2	Portability of ambient narratives	1
SE.3	Versioning of ambient narratives	2
SE.4	Modification of ambient narratives on location	1
SE.5	Debugging the behavior of ambient narratives	2
SE.6	Plug and write actuators	3
SE.7	Plug and write sensors	3

Table 3.2: List of functional requirements

4

Modelling Smart Environments

In this chapter we analyze and translate the functional retail requirements listed in section 3.5 into a formal description of the problem of reconstructing ambient intelligence from modular fragments. This step is needed in order to arrive at a machine readable representation of ambient narratives and a concrete working prototype system with authoring support that implements this interaction concept.

The rest of this chapter is organized as follows: Section 4.1 provides the definitions and notations that are used to formalize the problem and describe the proposed solution. In sections 4.2 and 4.3 the formal model is constructed based upon the elicited requirements for the special case of a static set of fragments. Section 4.4 extends this formal model to include situations where the set of fragments dynamically changes over time. We end this chapter with a discussion of related work that intersects hypertext, interactive storytelling and ambient intelligence.

4.1 Definitions and notations

We first introduce the elements and attributes of the ambient narrative language model and provide formal descriptions of these elements and attributes. We make a difference between the complete language model and a core language model that implements a subset of the full language model which is easier to understand for designers and therefore simplifies the authoring environment. It is still sufficient to let designers recreate the intelligent shop window scenario introduced in section 3.3.

Ambient intelligence is to be seen as a mass customized information product that is delivered via the network through the collective user interface of the devices that surround people (Requirements AN.2 and AN.3) (see also discussion sections 4.5.1 and 4.5.2). We seek a *declarative* language model that concentrates on specifying the desired context-aware behavior (the 'intelligence') and that leaves the actual rendering and implementation of this behavior ('ambience' creation) to the individual devices and applications.

To model social scripts enhanced with ambient technology (Requirement AN.1) there are two elements that need to be described: the social script (situation) and the ambient effects (action) that augment this performance. An ambient intelligence fragment therefore consists of two parts, a situation description and an action description. The situation description can be a combination of physical context and narrative state restrictions (see also discussion section 4.5.3). The action specifies the actions that must be rendered over the devices(s) in the environment (see also discussion section 4.5.4).

Definition 4.1 (Beat)

A *Beat b* is a tuple $< b_{id}, b_{cl}, b_{ke}, pre, act >$ where b_{id} is the name of the beat, b_{cl} is the group of beats this beat belongs to, b_{ke} is a sequence of words that contains additional metadata set by the author, *pre* defines the *Preconditions* of the beat, *act* defines the *Action* on the devices in the environment that is to be executed when the preconditions of the beat hold.

To stay close to the dramaturgical theory of social life we use the term *beat* from drama to refer to an ambient intelligence fragment. In drama and film theory, a beat *is an exchange of behaviour in action/reaction*. Beat by beat these changing behaviours shape the turning of a scene [McKee, 1997]. A sequence of beats results in a scene that can be defined as *an action through conflict in more or less continuous time and space that turns the value-charged condition of a character's life on at least one value with a degree of perceptible significance.* [McKee, 1997]. In dramatic plays and stories, every scene creates a meaningful change in the life situation of a character. To create meaningful experiences story writers deliberately place characters in conflict situations because these conflict situations force characters to make choices that will reveal their true characters and create the meaning of the story. Not all events in our everyday life seem dramatic, but a 'good' user scenario created by an ambient intelligence designer typically has the form of a dramatic story. This is clear for certain ambient narrative genres like the entertainment examples of Chapter 2 but can also be seen in more functional ambient narratives. Consider the following example: The user puts white clothes in the washing machine, but forgets there was a new red t-shirt in the set of clothes. The intelligent washing machine detects the tag on the t-shirt and signals the user to remove the t-shirt. The user removes the t-shirt. In this typical ambient intelligence scenario a conflict situation is created (unwashed t-shirt in set of clothes to be washed) that builds up to a climax (placed inside the washing machine) but then suddenly the scene turns (the intelligent washing machine recognizes the t-shirt) and leads to a conclusion (user removes the t-shirt). The dramaturgical theory of social life will therefore be taken literally here.

The activation and deactivation of beats is not only controlled by setting the preconditions of a beat, but also by event handlers or trigger objects that can be set (added) or retracted (removed) in a trigger list (Definition 4.7).

Definition 4.2 (Trigger)

A *Trigger* t is a tuple $< t_{id}, t_{be}, pre, l >$ where t_{id} is the identifier of the trigger object, $t_{be} \in (add, remove)$ defines the operation on the trigger list and *pre* the preconditions that should be satisfied before the *Link* l is traversed.

A trigger is a context-aware link found in physical hypermedia systems like e.g. [Hansen, Bouvin, Christensen & Grønbæk, 2004; Millard, Roure, Michaelides & Thompson, 2004]. Similar to a beat each trigger has a preconditions element (Definition 4.10). The trigger's precondition describes the social script that must be satisfied before the link element is traversed.

Definition 4.3 (Link)

Each *Link* l is a tuple $< l_{id}, l_{to}, l_{fr} >$ where l_{id} is the identifier of the link object, l_{to} the addition beat query, l_{fr} the removal beat query.

Each link object describes an operation on the set of beats that is currently active. A link object can cause beats to be removed and/or added at the same time to model complex operations on the set of active beats. All links in an ambient narrative are context-dependent because any context situation that can be expressed in the model is a situation that may cause a link to be traversed.

4.1.1 Ambient narrative

An ambient narrative can now be viewed as a graph consisting of nodes (beats) and edges between these nodes (triggers). Because beat nodes are hierarchic (tree-like structure), they can be represented as XML documents, as shown in

Figure 4.1.

```
<beat id="id" ... >
<pre>
  <stage id="stage" ... >
   <performance id="performance" ... >
   ...
   </performance>
  </stage>
</pre>
<action>
  <init> ... </init>
  <main> ... </main>
  <final> ... </final>
</action>
</beat>
```

Figure 4.1: Beat as XML document

Mass customization of ambient intelligence fragments (Requirement AN.2) is supported by the beat and trigger concepts. By performing in the environment, by interacting with the ambient narrative, actors cause beats to become activated or deactivated by implicitly or explicitly traversing links between beats and this in turn will affect the actions that will be rendered over the devices. We can use XPath path expressions [XPath, 2008] in the following to refer to navigation in those trees. The specification of which beats should be added or removed in a link object is expressed by such an XPath path expression: Figure 4.2 illustrates a link that will remove all beats in the active set that are a members of the DEMO class with a walking performance and add a single beat named DEMO1 to the set.

```
<link id="link123" to="//beat[@id='demo1']"
 from="//beat[@class='demo']/pre/stage/performance[id='walking']"/>
```

Figure 4.2: XPath path expressions in a link node

We are primarily interested in the interaction of an ambient narrative with the environment over time. Before operations on ambient narratives can be discussed in section 4.3 and 4.4 it is necessary to define the state of an ambient narrative so that we can define operations on ambient narratives in terms of state changes.

Definition 4.4 (Context)

The *Context Ctx* is defined as a list of *Stage* elements (Definition 4.11), $Ctx = (st_1, ..., st_n)$. Stage elements and their children have absolute values derived from real-world sensors.

The state of an ambient narrative first of all depends on the state of the physical environment. By performing, by acting out social scripts in the physical environment, people implicitly cause beats to be activated and deactived that will in turn change the behavior of the electronic devices surrounding them. Context is described by a list of stage elements with absolute sensor-derived values in the child Condition (Definition 4.12) elements and attributes for easy comparison of stage objects in beats and physical context.

Definition 4.5 (Story memory)

The *Story memory Mem* is the set of story values, $Mem = (sv_1, ..., sv_n)$ known by the ambient narrative system.

The ambient narrative state is determined by session state. Explicit user interaction with applications can result in session state variables referred to as story values (Definition 4.20) being set or retracted. In addition, beats themselves can modify story values to affect the beat sequencing process in the action part of the beat. The story memory contains the current values of these story values.

Definition 4.6(Beat set)

The *Beat set B* is the list of all *Beat* elements known by the ambient narrative system, $B = (b_1, ..., b_n)$.

The ambient narrative itself is a set of beats. In mass customization environments this beat set is fixed and defined in advance by the author of the ambient narrative, in co-creation environments (Requirement AN.4) the set of beats changes over time due to explicit interaction (e.g. using an authoring tool) or implicit interaction (e.g. device action starts a software agent that modifies this beat set).

Definition 4.7 (Trigger list)

The *Trigger list T* is the list of *Trigger* elements that are tracked by the system, $T = (t_1, ..., t_n)$. The set of beats associated to a trigger list T_B is a subset of the total beat set, $T_B \subset B$.

The trigger list restricts the set of beats that can become active at any time. Only the preconditions of the triggers in the trigger list will be checked. If the preconditions of a trigger are satisfied, the corresponding link will be traversed, resulting in beats that become activated and deactivated. The trigger list acts as a filter on the beat set.

Definition 4.8 (Active beat set)

The *Active beat set AB* is the subset of the *Beat set* that is currently active, $AB \subset B$. The active beat set is a subset of the beats that can be triggered by the trigger list, $AB \subset T_B$. T_{AB} defines the triggers of the active beat set.

The beats that have been retrieved by links/queries and that satisfy the current context and session state are stored in the active beat set. This active beat set jointly determines the intelligent behavior of the electronic devices that surround people in the ambient narrative environment.

Definition 4.9 (Ambient narrative)

The state of an *ambient narrative an* is described by the tuple $< Ctx, Mem, B, T, AB >$.

The state of an ambient narrative can then be defined by the state of the logical symbolic context (Definition 4.4), session state in the story memory (4.5), beats in the beat set (4.6), trigger list (4.7) and beats in the active beat set (4.8).

In the remainder of this section the formal definitions of the *Precondition* and *Action* section of a beat are given.

4.1.2 Preconditions

The preconditions of a beat specify the restrictions placed on the social scripts that are performed in the environment.

Definition 4.10 (Preconditions)

The *Preconditions pre* element contains the social script and consists of a single *Stage* element s, $pre = (s_1)$.

As the ambient narrative system should be able to react to implicit context changes (Requirement RT.1) specified in Table 3.1, explicit user feedback (Requirement RT.2) and session state (Requirement RT.3), these requirements should be reflected in the preconditions part of a beat. The more precisely these restrictions can be formulated in the model, the more precisely an author can specify the event that triggers the action.

Definition 4.11 (Stage)

Each *Stage s* is a tuple $< s_{id}, P, C, ti >$ where s_{id} is the symbolic name of the place where the social script is performed, P a set of *Performance* elements, $P = (p_1, ..., p_n)$, C a set of environmental *Condition* elements, $C = (c_1, ..., c_n)$

and ti a *Time* restriction element.

Because most social scripts are tied to a particular place and time, each preconditions object has one stage object. If this stage object is completely empty, there are no restrictions on activation. On a stage, zero or more activities (performances) can take place at the same time. Examples of performances are presence, walking, running, sleeping, driving. A stage does not necessarily have to be identical to a physical location, a stage can extend across several physical locations to support telepresence type of applications. A stage can furthermore have a set of environmental condition restrictions and restrictions set on the allowed time interval.

Definition 4.12 (Condition)

A *Condition* c is defined by the tuple $< c_{id}, c_{val}, c_{min}, c_{max} >$ where c_{id} is the name of the environmental condition, c_{val} the current value, c_{min} is the minimum allowed value and c_{max} is the maximum allowed value. Values are in real numbers.

Environmental condition range restrictions are described by a name to indicate the type of environmental condition (e.g. temperate, humidity) and a minimum and maximum allowed value.

Definition 4.13 (Time)

A *Time* element ti is a tuple $< ti_{d}t, ti_{min-w}, ti_{max-w}, ti_{min-d}, ti_{max-d}, ti_{min-t}, ti_{max-t} >$ where $ti_{d}t$ is the current date and time [1], ti_{w} is the current week number, ti_{d} the current day number, ti_{t} current time, ti_{min-w} an integer that defines the lowest allowed week number (between 1 and 52) in a year, ti_{max-w} the highest allowed week number (between 1 and 52), ti_{min-d} the first day (between 1 and 7) and ti_{max_d} the last day of a week between 1 and 7), ti_{min-t} the start time (HH:MM:SS), and ti_{max-t} the stop time (HH:MM:SS).

Some retailers change the collection four times a year, others every day of the week. In the intelligent shop window scenario, after the shop is closed a different set of beats could be active than during the day. To model these situations, time interval restrictions are allowed on week number, day of the week, and time of day.

[1]in (YYYY-MM-DDThh:mm:ss.sTZD) [W3C, 2008]

Definition 4.14 (Performance)

Each *Performance p* is a tuple $< p_{id}, p_{cl}, p_{du}, p_{no}, A, Pr, sc >$ where P_{id} is the name of the performance, P_{cl} is the class of performances this performance belongs to, P_{du} is the minimum duration of the performance (HH:MM:SS), P_{no} a boolean value that tests on existence or non-existence, A a set of *Actor* elements, $A = (a_1, ..., a_n)$, Pr a set of *Prop* elements, $Pr = (pr_1, ..., pr_n)$, sc a *Script* element.

A performance typically involves one or more people (actors) and one or more objects and devices (props). A shopper (actor) can for example be standing (performance) in front of a shop window display (prop). An actor or prop can be involved in several performances at the same time. For example, an actor can be present in the same room with an object while at the same time looking at another object.

Each performance may have a name (e.g. selecting item x from the shelf) that can be part of a larger group (e.g. selecting an item from the shelf). An author can set a restriction on the class of the performance to write only one beat for a class of activity instances. An actor can also be performing a certain activity for a period of time (e.g. standing still) before we want to trigger a device action (e.g. product is highlighted). Finally there can also be situations (e.g. shop closes) where we are interested in testing on the non-existence of a performance (e.g. person inside). Besides restrictions on the (physical) context situation, the system should also respond to explicit user feedback (Requirement RT.2) and changes in session state (Requirement RT.3). The author can use a script object to describe such situations (Definition 4.19).

Definition 4.15 (Actor)

An *Actor a* is defined by the tuple $< a_{id}, a_{ro}, a_{no}, a_{min-c}, a_{max-c}, po, o >$ where (a_{id} is the name of the actor, a_{ro} is the role of the actor in the performance, a_{no} is a boolean value that tests on existence or non-existence, a_{min-c} is the minimum number of actors allowed, a_{max-c} is the maximum number of actors allowed, po a *Position* element and o an *Orientation* element.

An actor is an entity that plays an active role in the social script and can be a human in the role of a shopper or shop employee, for example, or an animal in the role of a pet or a virtual agent that represents a real user. Authors should be able to set restrictions on the name of the actor (e.g. John), its role (e.g. shopper), whether the actor must be involved in this performance or not (e.g. shopper not present in front of shop window) and what the minimum and maximum number of allowed instances of this role of actors is in the performance (groups). To describe restrictions on the position and orientation of an actor or group of actors the author can use position and orientation objects respectively.

Definition 4.16 (Prop)

Each *Prop pr* is a tuple $< pr_{id}, pr_{no}, pr_{min-c}, pr_{max-c}, pr_{ca}, po, o >$ where (pr_{id} is the unique identifier of the prop, pr_{no} is a boolean value that tests on existence or non-existence, pr_{min-c} is the minimum number of props allowed, pr_{max-c} is the maximum number of props allowed, pr_{ca} is the set of capabilities of this device, po a *Position* element and o an *Orientation* element.

A prop is an entity that plays a passive role in the social script and is typically a device or object that is being used or consumed in the performance. Props can be tangible objects such as items for sale in a shop or devices that render actions. The devices associated with the device action commands specified in the action part of a beat are logically also included as props in the preconditions part of that beat. Authors should be able to set restrictions on the identifier of the prop (e.g. SHOPWINDOWAPP1), its capabilities (e.g. FLASHAPP), whether the prop must be involved in this performance or not, and what the minimum and maximum number of allowed instances of this type of props is (groups). To describe restrictions on the position and orientation of a prop or group of props the author can use position and orientation objects respectively.

Definition 4.17 (Position)

The tuple $< po_{co}, po_{anc}, po_{anc-a}, po_{anc-d}, po_{max-d}, po_{min-d}, po_{ang}, po_{ang-o}, Co >$ describes a *Position* element po. po_{co} represents the coordinates of the object. Coordinates are specified in a cartesian or spherical coordinate system. po_{anc} (anchor) refers to an *Actor* or *Prop* object identifier relative to this object. po_{anc-a} is the angle to the translated anchor point, po_{anc-d} is the distance to the translated anchor point, po_{max-d} is the maximum allowed distance from the (translated) anchor point, po_{min-d} is the minimum allowed distance from the (translated) anchor point, po_{ang} is the angle of the (translated) anchor point, po_{ang-o} is the offset angle from the (translated) anchor point and Co defines a set of polygon corner coordinates to mark an absolute area. If absolute corner coordinates are set, all other attributes are not used.

Restrictions on the position of actors and props can be absolute (e.g. object must be within certain absolute coordinates) or relative to another actor or

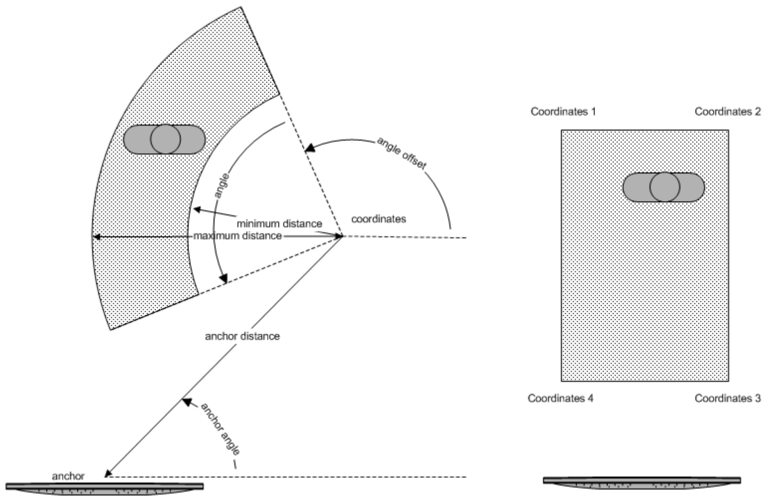


Figure 4.3: Restrictions on actor/prop positions

prop (e.g. object X should be at maximum two meters from object Y) and in XYZ or GPS coordinates. Figure 4.3 visually illustrates position restrictions that can be set. The gray area marks the area relative to the LCD panel where an actor needs to be present. Instead of relative coordinates (left) the author can also use absolute coordinates (right).

Definition 4.18 (Orientation)

Each *Orientation o* is a tuple $< o_{max-x}, o_{min-x}, o_{max-y}, o_{min-y}, o_{max-z}, o_{min-z} >$. o_{max-x} is the maximum allowed rotation around the x-axis (pitch), o_{min-x} is the minimum allowed rotation around the x-axis, o_{max-y} and o_{min-y} specify the allowed interval allowed for the y-axis of rotation (roll), o_{max-z} and o_{min-z} specify the allowed interval allowed for the z-axis (yaw).

The allowed orientation intervals of actors and props are specified in degrees and can be set for the x-axis (pitch), y-axis (roll) and z-axis (yaw).

Definition 4.19 (Script)

Each *Script sc* is a set of *Story value* elements, $Sc = (sv_1, ..., sv_n)$.

Each script object contains a number of story value objects. Story value objects can be modified in the action part of a beat and checked in the precondition part to test if certain beats are already active or not, for example, and thereby support reasoning about session state (Requirement RT.3) . Applications activated through device action commands in the action part of a beat may also send modified story values to the ambient narrative system, allowing reasoning about user feedback (Requirement RT.2).

Definition 4.20 (Story value)

A *Story value sv* is a tuple $< sv_{ac}, Sv_{id}, sv_{va}, sv_{min-v}, sv_{max-v}, sv_{no} >$ where sv_{ac} is the operation on the *Story memory* ($sv_{ac} \in (test, add, remove)$), sv_{id} is the name of the story value, sv_{va} is the current value of the story value, sv_{min-v} is the minimum allowed value, sv_{max-v} is the maximum allowed value, sv_{no} tests on the existence or non-existence.

Restrictions on story values can be set in the preconditions part to check whether a particular story value is within/outside a certain interval. In the action part of a beat the author of an ambient narrative can modify the story memory (Definition 4.5) and change the current value of a story value. Both actors and props may have scripts (profiles) with embedded story value name and value pairs associated to them to indicate actor and prop preferences. As soon as an actor or prop with a script is recognized by the system, the script is parsed and the story values are inserted into story memory and from then on influence the beat sequencing process.

4.1.3 Action

The action part of a beat describes the effect on the environment when all the restrictions set on the social script in the preconditions part are satisfied.

Definition 4.21 (Action)

Each *Action* element *act* is a tuple $(ac_{ty}, ac_{ta}, ac_{pr}, i, m, f)$ where ac_{ty} is the presentation markup language, ac_{ta} is the machine address of the rendering engine, ac_{pr} is a preview media element for visualization purposes, i is an *Init* element, m is a *Main* element and f a *Final* element.

The action part consists of three subparts: The *Init*, *Main* and *Final* elements that contain device action commands to change the ambience and operations to modify the state of the ambient narrative. We designed a declarative remote procedure call language (Definition 4.22) that allows the ambient narrative author to specify timed actions on devices to deal with a variety of devices with different capabilities and diverse applications running on those devices (Requirement RT.4). Other declarative presentation language standards such as the Hypertext Markup Language [HTML, 2008] or Synchronized Multimedia Integration Language [SMIL, 2008] may also be used to e.g. describe the layout of graphics and text or timing and synchronization of media elements by setting the $(ac_{ty}$ attribute. Finally, the action element can contain a preview image or media clip that shows a preview of the ambience generated by the effect.

Definition 4.22 (Action script)

An *Action script d* is a simple presentation language markup that consists of a tuple $< d_{id}, d_{be}, Da >$ where d_{id} is the identifier of the action script object, $d_{be} \in (add, remove, retract)$ is the behavior of the action script and Da a list of *Device action command* elements, $Da = (da_1, ..., da_n)$.

A set of device action commands is grouped in an action script object. Each action script object has a unique name. A behavior attribute can be used to add, remove and retract the script with the given name. In case of a remove operation the script and all still scheduled device actions with it will be retracted from the rendering engine. When the retract behavior is specified all the devices that have already been set to a particular value will be set back to their default value and the other device actions are canceled. If an action script has, for example, already set four lamps to a red colour by the time the script is retracted and three lamps are still scheduled to turn to a blue colour, the device actions for the three lamps are canceled and the four red lamps will be set to their default value (e.g. switched off).

Definition 4.23 (Device action command)

Each *Device action command da* is a tuple $< da_{st}, da_{de}, da_{ac}, da_{zo} >$ where da_{st} is the start time of the action, da_{de} is the symbolic name of the target device, da_{ac} the value that is send to the target device at the specified start time and da_{zo} the z-order.

Each device action command is basically a remote procedure call on a device. The procedure call itself is in the form of a string that is passed on in a value attribute. The device action command provides an abstraction that hides any device specific attributes. The result of this device action can be that a lamp changes colour, a video clip starts on a projection display or that a software application changes its state depending on the target device and the specified value. A device action command can be executed immediately after it has been forwarded to the rendering engine or with a delay to allow for timed actions. The priority of the device action can be set with a z-order attribute. If there are multiple device commands scheduled for one device, the device command with the lowest z-order will be rendered in the background and the one with the highest z-order on top. How this priority scheme is implemented is left to the device itself (e.g. overriding behavior, overlaying images on top of each other, mixing sounds with different volumes).

XPath path expressions can also be used to replace fixed device action command values by a query on an action content database to support customization of device actions based on context and session state and personalization of action scripts. Figure 4.4, e.g., specifies a device action command that contains a query for an image URL that has been created by the actor on stage DESK2 in the context database. This XPath expression is resolved in an image URL when the beat is initialized. The advantage of using action script templates is that they require fewer beats to be written and fewer triggers to keep track of.

```
<command start="+00:00:00" target="display1" z-order="1000"
 value="//content[@type='image' and
 @author='//ctx/stage[@id='desk2']/*/actor/@id]/@url]'"/>
```

Figure 4.4: XPath path expressions in a device action command node

Definition 4.24 (Init)
An *Init* I is an unordered list (Sv, T, d) where Sv is a set of *Story value* elements, $Sv = (sv_1, ..., sv_n)$, T a set of *Trigger* elements, $T = (t_1, ..., t_n)$ and d a presentation language markup that is sent to the rendering engine for interpretation.

The initialization or 'init' element of a beat action is executed immediately after the beat has become active and before the main element is processed. During this initialization step story values can be set or modified that affect the beat sequencing process. For example a beat can set a story value during initialization that is checked by another beat in its preconditions. In this way authors can define dependencies between ambient narrative fragments. In addition to setting story values (Definition 4.20), an initialization can set triggers (Definition 4.2) to define context-dependent beat queries that affect the beat sequencing process. Presentation markup specified in an 'init' object is forwarded to the rendering engine for interpretation, but in most cases the device actions will be specified in the main element.

Definition 4.25 (Main)

Each *Main* element m is an unordered list (d) where d is a presentation language markup that is sent to the rendering engine for interpretation.

The main part of a beat action is executed after the beat initialization has been completed and contains presentation markup that will be sent to a rendering engine for interpretation. This rendering engine only controls the timing and synchronization of actions, the actual rendering is left to the devices, analogous to a Web browser that provides a similar abstraction layer.

Definition 4.26 (Final)

A *Final* F is an unordered list (Sv, T, d) where Sv is a set of *Story value* elements, $Sv = (sv_1, \ldots, sv_n)$, T a set of *Trigger* elements, $T = (t_1, \ldots t_n)$ and d a presentation language markup that is sent to the rendering engine for interpretation.

The final element is processed when the preconditions of a currently active beat are no longer satisfied and it therefore has been scheduled for removal. The final part can be used to remove story values that are no longer used or retract triggers that listen for events that are no longer of interest. To avoid presentations from continuing to run after a beat has been removed, device action commands in the presentation markup can be used to retract the script and revert back to the default values of the devices (e.g. shop window display mode is blanked) or overriding the behavior (e.g. by sending a 'standby' command to a device).

This completes the list of definitions and notations of the ambient narrative language model.

4.1.4 Core language model

The more precisely an author can set restrictions on the activation of beats, the more detailed situation descriptions can be formulated. This increased functionality comes, however, at a cost. First, the system needs to keep track of more situations that can take place in the environment (and trigger events) which may reduce the system's real-time performance. Second, the cognitive load of the authoring environment on the user is increased. Third, certain events are hard to detect (e.g. activity and emotion detection) and therefore difficult to implement given the current state of the art in sensor technology. For these reasons the functionality offered by the run-time and authoring environment of the intelligent shop window is a subset of the functionality that can be expressed in the language model. This core language is sufficient to let end-users describe and program the intelligent shop window scenario in Section 3.3 while staying true to the ambient narrative concept. The design decisions and implementation of the end-user programmable shop window system are discussed in detail in Chapter 6.

Table 4.1 provides an overview of the introduced definitions and notations in the order they were presented and marks whether these definitions are also part of the core language model that can be modified by the authoring environment. Lowercase symbols indicate individual elements, uppercase symbols refer to sets of those elements. For attributes of the corresponding elements we refer to the individual definitions.

symbol	**description**	**core language?**
b	beat	*
t	trigger	*
l	link	* (beat links only)
Ctx	context	*
Mem	story memory	*
B	beat set	*
T	trigger set	*
AB	active beat set	*
an	state of ambient narrative	*
pre	preconditions	*
s	stage	*
c	condition interval	
ti	time interval	
p	performance	* (presence detection only)
a	actor	* (a_{id},a_{ro},a_{no})
pr	prop	* (a_{id},a_{ca},a_{no})
po	position	* (absolute values only)
o	orientation	
sc	script	*
sv	story value	*
act	action	* (no action customization)
d	action script	*
da	device action	*
i	init	* (no action script)
m	main	*
f	final	* (no triggers)

Table 4.1: List of definitions and notations

In the preconditions part, condition, orientation and time interval elements are excluded because environmental sensor input data, orientation of objects and date and time of day are not used in the intelligent shop window scenario. Furthermore, we limit ourselves to presence detection of people and objects in certain areas. For actors, end-user tests are allowed on name, role and presence attributes, for props on their unique id, capability and presence attributes. In terms of position restrictions authors can specify absolute stages and indicate the presence and/or absence of actors and/or props.

In the action part of a beat the core language model does not allow authors to place action scripts in beat initialization section and define trigger list operations in the main and final sections of a beat. Customized device actions (XPath expressions in device action values) are also not included in the core language model.

4.2 Precondition checking

To determine whether all the restrictions set on the social script in the precondition part of a beat or trigger are satisfied, its precondition fragment has to be matched against the current context. Formally, this problem can be stated as a subtree matching problem. Subtree matching is the problem of finding all the matching occurrences of a pattern tree in a subject tree as a subtree [Cserkuti, Levendovszky & Charaf, 2006]; in this case the precondition fragment is the pattern tree, the context the subject tree.

To distinguish between context and precondition trees we use the superscript c for context and p for precondition nodes. Subscripts refer to attributes, e.g. a^p_{id} refers to the *id* attribute of an actor node a as part of the precondition p. XPath path expressions are used for node tests, e.g. s^p/p^p refers to a performance node that is a child of a stage node, both in the precondition tree. XPath path expressions may also be used as alternative syntax to select attributes, e.g. $a^p/@id$ refer to the *id* attribute of an actor node a as part of the precondition p ($a^p_{id} \equiv a^p/@id$).

Each precondition tree *pre* has a stage child node with a specific id attribute. The value of this id attribute must occur in at least one of the stage nodes of the context tree *ctx* to satisfy the stage test, $StageTest(pre, ctx)$:

$$s^p \in pre/s \Rightarrow \exists s^c \in ctx/s \wedge SC(s^p, s^c) \qquad (4.2.1)$$

The stage check of two stage nodes can be defined as follows:

$$SC(s^p, s^c) = \begin{cases} 1, & \text{if } s^p_{id} = s^c_{id} \\ 0, & \text{otherwise} \end{cases}$$

If a precondition tree contains a time restriction on a particular stage, there should be a time node in the context tree that satisfies the time test, $TimeTest(pre, ctx)$:

$$s^p/ti^p \in pre/s \Rightarrow \exists s^c/ti^c \in ctx/s \wedge TC(ti^p, ti^c) \tag{4.2.2}$$

The comparison of two time nodes is described by:

$$TC(ti^p, ti^c) = \begin{cases} 1, & \text{if } ti^p_{min-w} \leq week(ti_{dt}) \leq ti^p_{max-w} \wedge \\ & \quad ti^p_{min-d} \leq day(ti_{dt}) \leq ti^p_{max-d} \wedge \\ & \quad ti^p_{min-t} \leq time(ti_{dt}) \leq ti^p_{max-t} \\ 1, & \text{if } ti^p = \emptyset \\ 0, & \text{otherwise} \end{cases}$$

If a precondition tree contains an environmental condition restriction on a particular stage, there should be a condition node in the context tree that satisfies the condition test ($ConditionTest(pre, ctx)$):

$$s^p/c^p \in pre/s \Rightarrow \exists s^c/c^c \in ctx/s \wedge CC(c^p, c^c) \tag{4.2.3}$$

The comparison of two condition nodes can be defined as follows:

$$CC(c^p, c^c) = \begin{cases} 1, & \text{if } c^p_{min} \leq c^c_{val} \leq c^p_{max} \\ 0, & \text{otherwise} \end{cases}$$

A context tree needs to satisfy the performance test for all performance restriction nodes set in the precondition tree, $PerformanceTest(pre, ctx)$:

$$s^p/p^p \in pre/s \Rightarrow \exists s^c/p^c \in ctx/s \wedge PC(p^p, p^c) \tag{4.2.4}$$

The comparison of two performance nodes is defined as follows (if p^p_{no} not defined use $p^p_{no} = 0$):

$$PC(p^p, p^c) = \begin{cases} p^p_{no} \oplus 1, & \text{if } (p^p_{cl} = p^c_{cl} \vee p^p_c l = \emptyset) \wedge \\ & \quad (p^p_{id} = p^c_{id} \vee p^p_{id} = \emptyset) \wedge \\ & \quad (p^p_{dur} \leq p^c_{dur}) \\ p^p_{no} \oplus 0, & \text{otherwise} \end{cases}$$

If a precondition tree contains an actor restriction node as child of a stage and performance node, there should be an actor node in the context tree that satisfies the actor test, $ActorTest(pre, ctx)$:

$$\forall s^p/p^p \in pre/s \Rightarrow \exists s^c/p^c \in ctx/s \wedge AC(p^p, p^c) \tag{4.2.5}$$

The comparison of two actor nodes is defined as follows (if a^p_{no} not defined use $a^p_{no} = 0$):

$$AC(p^p, p^c) = \begin{cases} a^p_{no} \oplus 1, & \text{if } \forall a^p \in actor(p^p) : ARC(p^p, p^c) \wedge AIC(p^p, p^c) \\ a^p_{no} \oplus 0, & \text{otherwise} \end{cases}$$

The number of actors in a particular role in the context tree should be within the minimum and maximum set values in the precondition tree ($A_x = p^p/a[@ro =' x']$):

$$\begin{aligned} ARC(p^p, p^c) =& \forall x \in A_x : \\ & \sum_{a_{min} \in A_x} a_{min} \leq \mid p^c/a[@ro =' x'] \mid \leq \sum_{a_{max} \in A_x} a_{max} \end{aligned}$$

Names of actors specified in the preconditions should also exist in the context:

$$AIC(p^p, p^c) = \forall a^p_{id} \in p^p_A \exists a^c_{id} \in p^c_A : a^p_{id} = a^c_{id}$$

If a precondition tree contains a prop restriction node as child of a stage and performance node, there should be a prop node in the context tree that satisfies the prop test, $PropTest(pre, ctx)$:

$$\forall s^p/p^p \in pre/s \Rightarrow \exists s^c/p^c \in ctx/s \wedge PrC(p^p, p^c) \tag{4.2.6}$$

The comparison of two prop nodes is similar to the actor node comparison and can be defined as follows (($pr^p_{no} = 1$ if $pr^p_{no} = \emptyset$):

$$PrC(p^p, p^c) = \begin{cases} pr^p_{no} \oplus 1, & \text{if } \forall pr^p \in prop(p^p) : \\ & PrRC(pr^p, pr^c) \wedge PrIC(pr^p, pr^c) \\ pr^p_{no} \oplus 0, & \text{otherwise} \end{cases}$$

The number of props with a particular capability should be within the minimum and maximum set values for that capability ($Pr_x = p^p/pr[@cap = contains('x')]$):

$$PrRC(p^p, p^c) = \forall x \in p^p/pr[@cap = contains('x')] : \\ \sum_{pr_{min} \in Pr_x} \leq | \, p^c/pr[@cap = contains('x')] \, | \leq \sum_{pr_{max} \in Pr_x} pr_{max}$$

Names of props specified in the precondition tree should also exist in the context tree.

$$PrIC(p^p, p^c) = \forall pr^p_{id} \in p^p_{Pr} \exists pr^c_{id} \in p^c_{Pr} : pr^p_{id} = pr^c_{id}$$

If a precondition tree contains an orientation node, there must be a matching orientation node in the context tree, $OrientationTest(pre, ctx)$:

$$\begin{aligned} &\forall s^p/p^p/a^p \in pre/s \Rightarrow \exists s^c/p^c/a^c \in ctx/s \wedge OC(a^p_o, a^c_o) \\ &\forall s^p/p^p/pr^p \in pre/s \Rightarrow \exists s^c/p^c/pr^c \in ctx/s \wedge OC(pr^p_o, pr^c_o) \end{aligned} \tag{4.2.7}$$

The values for rotation around the x-axis (pitch), y-axis (roll), and z-axis (yaw) in the context orientation node must be within the specified intervals by the precondition orientation node.

$$OC(o^p, o^c) = \begin{cases} 1, & \text{if } o^p_{min-x} \leq o^c_{val-x} \leq o^p_{max-x} \wedge \\ & \quad o^p_{min-y} \leq o^c_{val-y} \leq o^p_{max-y} \wedge \\ & \quad o^p_{min-z} \leq o^c_{val-z} \leq o^p_{max-z} \\ 1, & \text{if } o^p = \emptyset \\ 0, & \text{otherwise} \end{cases}$$

If a precondition tree contains a position node, there must be a matching position node in the context tree, $PositionTest(pre, ctx)$:

$$\begin{aligned} &\forall s^p/p^p/a^p \in pre/s \Rightarrow \exists s^c/p^c/a^c \in ctx/s \wedge POC(a^p_{po}, a^c_{po}) \\ &\forall s^p/p^p/pr^p \in pre/s \Rightarrow \exists s^c/p^c/pr^c \in ctx/s \wedge POC(pr^p_{po}, pr^c_{po}) \end{aligned} \tag{4.2.8}$$

po^p can be an absolute or relative restriction on the position of an actor or prop. If po^p is absolute, the position (coordinates) of the object in the preconditions must be exactly the same as the current value for that object in the context tree. If po^p is relative, the current position of that object in the context tree must be contained by the area specified by po^p. po^c can be absolute or relative. A relative context position introduces uncertainty in the exact position of a sensed object and therefore leads to an inclusive answer as to whether the sensed object is in inside the area specified by a relative precondition position unless the relative context position is fully inside the relative precondition position.

$$POC(po^p, po^c) = \begin{cases} 1, & \text{if } inside(po^p, po^c) \\ 0, & \text{otherwise} \end{cases}$$

Definition 4.27 (Precondition context valid)

The context constraints in a precondition tree are met, $ContextCheck(pre, ctx)$, if constraints 4.2.1, 4.2.2, 4.2.3, 4.2.4, 4.2.5, 4.2.6, 4.2.7 and 4.2.8 are all satisfied.

Next to restrictions on the physical context, a precondition tree can also set constraints on story values to react on the internal session state of an ambient narrative. All story value restrictions in a precondition tree must be satisfied in the story memory:

$$\forall sv \in pre/*/sc \Rightarrow \exists m \in Mem : sv_{id} = m_{id} \wedge SVC(sv, m) \tag{4.2.9}$$

The comparison of two story values is defined as follows (($sv_{no} = 1$ if $sv_{no} = \emptyset$:

$$SVC(sv, m) = \begin{cases} sv_{no} \oplus 1, & \text{if } sv_{min} \leq m_{val} \leq sv_{max} \\ sv_{no} \oplus 0, & \text{otherwise} \end{cases}$$

Definition 4.28 (Precondition story values valid)

The story value constraints in a precondition tree are met, $StoryMemoryCheck(pre, mem)$, if constraint 4.2.9 is satisfied.

Definition 4.29 (Preconditions valid)

The preconditions of a beat or trigger are satisfied if both context and story value constraints are met: $PreCheck(pre, ctx, mem) = ContextCheck(pre, ctx) \wedge StoryMemoryCheck(pre, mem)$.

4.3 Beat sequencing

At any moment in time, the state of an ambient narrative *an* is described by the tuple (Ctx, Mem, B, AB, T) (Definition 4.9). However, after each change in physical context ($Ctx \rightarrow Ctx'$) or change in session state ($Mem \rightarrow Mem'$) the state of the ambient narrative needs to be updated ($an \rightarrow an'$). This problem that we will refer to as *beat sequencing* will be defined in a formal way next and is implemented by the *ambient narrative engine* that manages the state of the ambient narrative and controls the distribution of action scripts to the rendering engine (Chapter 6).

The set of active beats and trigger list remains unchanged $(Ctx, Mem, B, AB, T) \rightarrow (Ctx', Mem', B, AB, T)$ if the preconditions of all active beats are still valid and no new triggers are activated after a change in physical context or session state:

$$\forall b \in AB : PreCheck(b/pre, Ctx', Mem') \tag{4.3.1}$$

$$\forall t \in T \setminus T_{AB} : \neg PreCheck(t/pre, Ctx', Mem') \tag{4.3.2}$$

$(Ctx, Mem, B, AB, T) \rightarrow (Ctx', Mem', B, AB', T')$ if one or more of the preconditions tests of an active beat no longer hold or if a new trigger is activated:

$$\exists b \in AB : \neg PreCheck(b/pre, Ctx', Mem') \tag{4.3.3}$$

$$\exists t \in T, t \notin T_{AB} : PreCheck(t/pre, Ctx', Mem') \tag{4.3.4}$$

If the precondition test of one of the active beats fails (Rule 4.3.3 is satisfied), the beat should be removed from the active beat set and checked for story memory, trigger list operations and action script that must be sent to the rendering engine (REMOVEACTIVEBEAT rule):

REMOVEACTIVEBEAT

$$\frac{\begin{array}{ccc} b, AB, Mem, Ctx, T & sc \in b/*/final/sc & sv \in b/*/final/sv \\ t \in b/*/final/t & \multicolumn{2}{c}{\exists b \in AB : \neg PreCheck(b/pre, Ctx, Mem)} \end{array}}{\begin{array}{cc} UpdateStoryMemory(sv, Mem) & UpdateTriggerList(t, T) \\ SendActionScript(sc) & AB' = AB \setminus b \end{array}}$$

If a new trigger is activated (condition 4.3.3 is true), the link must be traversed (LINKTRAVERSAL). XPath queries in the link should be executed on the beat set and the results added to or removed from the active beat set depending on the link attribute in the following way:

LINKTRAVERSAL

$$\frac{AB, Mem, Ctx, T \qquad \exists t \in T, t \notin T_{AB} : PreCheck(t/pre, Ctx, Mem)}{\begin{array}{c} \forall b \in t/l_{fr}(B) : RemoveActiveBeat(b, AB) \\ \forall b \in t/l_{to}(B) : AddActiveBeat(b, AB) \end{array}}$$

The add beat operation (ADDACTIVEBEAT) adds a new beat to the active beat set and checks for story memory, trigger list operations and action script that must be sent to the rendering engine:

ADDACTIVEBEAT

$$\frac{\begin{array}{c} b, AB, Mem, Ctx, T \qquad sv \in b/*/init/sv \\ t \in b/*/init/t \qquad sc \in b/*/init/sc \qquad PreCheck(b/pre, Ctx, Mem) \end{array}}{\begin{array}{c} UpdateStoryMemory(sv, Mem) \qquad UpdateTriggerList(t, T) \\ SendActionScript(sc) \qquad SendActionScript(b/*/main/sc) \\ AB' = AB \cup b \end{array}}$$

The update trigger list operation (UPDATETRIGGERLIST) adds or deletes a trigger in the trigger list depending on the trigger action:

UPDATETRIGGERLIST

$$\frac{t, T}{t_{ac} = \text{`}remove' : T' = T \setminus t \qquad t_{ac} = \text{`}add' : T' = T \cup t}$$

The update story memory operation (UPDATESTORYMEMORY) adds or deletes a story value in the story memory depending on the story value action:

UPDATESTORYMEMORY

$$\frac{sv, Mem}{\begin{array}{c} sv_{ac} = \text{`}remove' : RemoveStoryValue(sv, Mem) \\ sv_{ac} = \text{`}add' : AddStoryValue(sv, Mem) \end{array}}$$

The add story value operation (ADDSTORYVALUE) is formally defined by:

ADDSTORYVALUE

$$\frac{sv, Mem}{\begin{array}{c} \exists m \in Mem, m_{id} = sv_{id} : m'_{val} = m_{val} + sv_{val} \wedge Mem' = Mem \setminus m \cup m' \\ \neg \exists m \in Mem, m_{id} = sv_{id} : Mem' = Mem \cup sv \end{array}}$$

The remove story value operation (REMOVESTORYVALUE) can be described by:

REMOVESTORYVALUE

$$\frac{sv, Mem}{\exists m \in Mem, m_{id} = sv_{id} : Mem' = Mem \setminus m}$$

The send action script operation *SendActionScript* sends the specified action script to the rendering engine that controls the timing and synchronization of actions over the distributed devices. The device actions specified in the action script are allowed to influence the beat sequencing process by sending the result of their action or other important events as a story value change back to the ambient narrative engine via the rendering engine to influence the beat sequencing process. The downside of using story value commands embedded in device actions is that they are outside the specification of the ambient narrative and therefore make it difficult to trace errors. To circumvent this problem an author may choose to break a complex beat down into smaller simpler beats with more detailed preconditions and story value commands in the ‘init’ or ‘final’ section of these beats. The rendering engine is discussed in section 6.2.3.

After a *RemoveActiveBeat* or *AddActiveBeat* operation has been completed, rules 4.3.3 and 4.3.4 need to be reevaluated as the story memory and/or trigger list may have changed which can cause active beats to become invalid and new triggers to be activated. This process repeats until conditions 4.3.1 and 4.3.2 are both satisfied. A single change in context or story memory can therefore cause a whole chain of effects. We refer to a single iteration as a *beat sequencing cycle*.

Ordering

The order in which story values and triggers are set or retracted may influence the beat sequencing process. For example if in an ‘init’ or ‘final’ part of a beat, two story values have the same id and different actions, the sequence in which the story value changes are processed will determine the outcome. To prevent this we specify the following constraint (analogous for final):

$$\forall sv \in b/*/init/sv \Rightarrow \neg\exists x \in b/*/init/sv : sv_{id} = x_{id} \tag{4.3.5}$$

For triggers a similar rule applies (analogous for final):

$$\forall t \in b/*/init/t \Rightarrow \neg\exists y \in b/*/init/t : t_{id} = y_{id} \tag{4.3.6}$$

The order of the modifications on the active beat set can also affect the outcome of the beat sequencing process. If T_m is the set of triggers that is produced by the list of modifications on the active beat set and SV_m is the set of story values produced by this list, then constraint 4.3.5 and 4.3.6 need to be generalized to (4.3.7) to prevent this from happening during a beat sequence cycle.

$$\begin{aligned} &\forall sv \in SV_m \Rightarrow \neg\exists x \in SV_m : sv_{id} = x_{id} \\ &\forall t \in T_m \Rightarrow \neg\exists y \in T_m : t_{id} = y_{id} \end{aligned} \tag{4.3.7}$$

4.3.1 Example beat sequences

The combination of setting and retracting triggers and story values in beats can be used to create complex dynamic behavior in ambient narratives. To illustrate it is useful to look at some example beat sequences.

$$(b_0, b_1) \overset{mem:-sv1+sv2, l:-b_1,+b_2}{\longrightarrow} (b_0, b_2) \overset{mem:-sv2, l:-b_2,+b_3}{\longrightarrow} (b_0, b_3) \tag{4.3.8}$$

Sequence 4.3.8 represents a workflow scenario where three tasks (b_1, b_2, b_3) need to be performed in sequence. The active beat set starts with b_1 in the set. b_1 has a beat preconditions test on the existence of story value 'sv1' and b_1 sets a story value 'sv2' during initialization and a trigger that tests on the non-existence of story value 'sv1', which triggers a link that causes b_1 to be removed and b_2 to be added. b_2 tests on the existence of story value 'sv2'. If a device action of b_1 now causes the story value 'sv1' to be retracted, the trigger will fire and b_1 will be removed and b_2 added. b_2 has a beat preconditions test on the existence of story value 'sv2' and b_2 sets a story value 'sv3' during initialization and a trigger that tests on the non-existence of story value 'sv1', which will then trigger a link that causes b_2 to be removed and b_3 to be added. b_3 tests on the existence of story value 'sv2'. If a device action of b_2 now causes the story value 'sv2' to be retracted, the trigger will fire and b_2 will be removed and b_3 added.

$$(b_0, b_3, b_4) \overset{ctx:-time1, l:-b_3,-b_4}{\longrightarrow} (b_0) \tag{4.3.9}$$

Sequence 4.3.9 presents a scenario where a task stops two beats after a period of time. The active beat set starts with b_3 and b_4 in the set, both triggered through b_0. Both b_3 and b_4 do not have a time test, but b_0 has set a trigger with a precondition containing a time test that will cause b_3 and b_4 to be stopped if satisfied. If the context now changes and the time test is satisfied, the link is traversed and b_3 and b_4 are removed.

$$(b_0, b_5) \overset{ctx:-actor1, l:+b_6}{\longrightarrow} (b_0, b_5, b_6) \overset{mem:-sv6, l:-b_6,+b_7,+b_8,+b_9}{\longrightarrow} (b_0, b_5, b_7, b_8, b_9) \tag{4.3.10}$$

Sequence 4.3.10 illustrates a scenario where after a certain period of inactivity three new tasks are started. In this case one of the beats in the active beat set has set a trigger whose link will be traversed if an actor 'actor1' removes himself from the stage. In that case a beat b_6 is started that will set a trigger that will cause beats b_7, b_8, b_9 to activate if the story value 'sv6' is active. After a period of time, this beat removes itself by retracting story value 'sv6' in its final part.

$$(b_0, b_1) \xrightarrow{ctx:+prop1, l:-b_1+b_2} (b_0, b_2) \xrightarrow{ctx:+prop1, l:-b_2,+b_1} (b_0, b_1) \tag{4.3.11}$$

Sequences can also repeat themselves, 4.3.11 presents a scenario that will continue to cycle $b_1, b_2, b_1, b_2, ...$ until prop 'prop1' is removed from the stage.

$$(b_0, b_1) \xrightarrow{t_1=1} (b_0, b_2, b_3, b_4) \xrightarrow{t_2=2} (b_0, b_2) \tag{4.3.12}$$

In a single user environment or any other monotasking situation, beat sequences will typically take place after each other. In multi-user scenarios, M actors may involved in N performances simultaneously leading to more complex operations on the active beat set. Sequences 4.3.8 and 4.3.9 may partially overlap or be contained in each other as shown in 4.3.12 for example.

Sequences may also mutually influence each other, e.g. sequence 4.3.8 could retract a story value 'sv6' and in that case immediately cause beats b_7, b_8, b_9 to be added in sequence 4.3.10.

Mutually overlapping and interacting performances cannot be eliminated because they are part of everyday life; but make programming of ambient intelligent environments complex because the author has to take a holistic view that considers all user scenarios at the same time and their interactions. A number of techniques however can be employed to deal with this race condition 'problem':

- ◇ Precise preconditions: The more context tests are specified in a precondition, the less chance two unrelated beats will affect each other in unexpected ways. In the shop window scenario example beats specify all the props they need. If a device is then switched off, the beat will not be activated.
- ◇ Keep the size of the trigger list and story memory small: Triggers and story values that are no longer needed, may still activate links or affect the beat sequencing process. The larger the number of story values in memory, the more chance of influencing the beat sequencing process.
- ◇ Story value for each active beat: Each beat can set a story value that represents its name in its init part that is again retracted in its final part. This way each beat can test on the existence or non-existence of other beats. In small ambient narratives, the author can then simply state which other beats may or may not be active for a given beat.

4.4 Run-time authoring

During the beat sequencing process, the trigger list and/or active beat set may change as a result of context and/or story memory changes but the beat set itself remains fixed. In this section we formalize the problem of modifying the beat set at run-time to address our requirement of run-time modification of ambient intelligence fragments (Requirement A.4) for co-creation environments (see also discussion section 4.5.5). This problem can be viewed as follows: Given a change in beat set ($B \rightarrow B'$) determine the next state of the ambient narrative ($an \rightarrow an'$).

A beat set change requires at least two modifications: A trigger that is added/removed in an (existing) source beat, b_s in the beat set and a target beat, b_t that is added to or removed from the beat set.

In case of an add operation, there should be at least one source beat in the new beat set with a trigger that has a link with a 'to' query that can return the target beat:

$$\exists b, b_t \in B, b/*/l \Rightarrow b \neq b_t \wedge b_t \in l_{to}(B) \tag{4.4.1}$$

Rule 4.4.1 can temporarily be relaxed if an author decides to add a set of new target beats that are activated through a single source trigger for example. The author first adds the new target beats to the ambient narrative. The new beats are then unreachable from other beats. The author can make these beats reachable by adding a trigger to an existing source beat that can cause these beats to be activated.

When a new beat has been added to the beat set and constraint 4.4.1 is met, it is necessary to check whether the source beat is currently active or not. If the source beat is not part of the active beat set, no action has to be taken other than modifying the beat set, but if the source beat is active, the trigger list and active beat set have to be updated and checked for activation unless the trigger is in the final part of the source beat.

ADDBEAT

$$\frac{b_t, b_s, b'_s, t'_s, B, T \qquad b'_s/*/l \Rightarrow b'_s \neq b_t \wedge b_t \in b'_s/*/l_{to}(B)}{\begin{array}{c} B' = B \cup b_t \setminus b_s \cup b'_s \\ b_s \in AB : AB' = AB \setminus b_s \cup b'_s, t'_s \in b'_s/action/init/t : UpdateTriggerList(t'_s, T) \end{array}}$$

In case of a remove operation, there should be no triggers left in the modified beat set that have a 'fr' or 'to' query in the beat set that only returns the (removed) target beat in order to prevent broken links.

$$\forall b, b_t \in B', \neg\exists b/*/l \Rightarrow$$
$$(b_t \in l_{fr}(B) \wedge \mid l_{fr}(B) \mid = 1) \wedge (b_t \in l_{to}(B) \wedge \mid l_{to}(B) \mid = 1) \tag{4.4.2}$$

The beat removal operation of a source beat, b_s and target beat, b_t can then formally be described by the REMOVEBEAT rule. If both the source and target beat are not present in the active beat set, only the beat set has to be updated. If the source beat is present in the active beat set, the trigger list needs to be updated and the active beat set modified with the new value for the source beat. If the target beat is also present in the active beat set, the target beat must be removed from the active beat set.

REMOVEBEAT

$$\frac{\begin{array}{c} b_t, b_s, b'_s, B, T \\ \neg\exists l \in b'_s/*/l \Rightarrow (b_t \in l_{fr}(B) \wedge \mid l_{fr}(B) \mid = 1) \wedge (b_t \in l_{to}(B) \wedge \mid l_{to}(B) \mid = 1) \end{array}}{\begin{array}{c} B' = B \setminus b_t \setminus b_s \cup b'_s \\ b_s \in AB : t'_s \in b'_s/action/init/t : UpdateTriggerList(t'_s, T), AB' = AB \setminus b_s \cup b'_s \\ b_t \in AB : RemoveActiveBeat(b, AB) \end{array}}$$

Modifying an existing beat in the beat set can be seen as a REMOVEBEAT operation followed by a ADDBEAT operation.

4.5 Discussion

This section compares the formalized ambient narrative model introduced in the previous sections with earlier work in hypertext, interactive storytelling and mixed reality applications. The goal of this section is to reflect back on the choices and decisions that were taken to meet each of the functional ambient narrative concept requirements and indicate what the inspirations were for the formal model and why existing solutions were insufficient for our purpose.

4.5.1 Separation of ambience and intelligence by a network

To view ambient intelligence as an information product or service that is delivered to the devices surrounding the user we need to make a distinction between the device action that is rendered and how this device action is rendered. This separation is common in hypertext systems and is therefore also adopted here. We can map the ambient narrative model on the Dexter hypertext reference model [Halasz & Schwartz, 1994]. This mapping helps to understand how beats and the presentation descriptions inside the action section of a beat relate. From the point of view of the ambient narrative engine that sequences/reads beats (run-time layer), these beats (Definition 4.1) correspond to the components in the Dexter storage layer, whereas the triggers (Definition 4.2) refer to links in the Dexter model. The presentation markup inside the action (Definition 4.21) section of a beat is part of the within-component layer because it is not further interpreted. However, if we consider the rendering engine that interprets this presentation markup, this markup is the presentation specification between the run-time and storage layer. The advantage of this approach is that we can use different Dexter-based presentation systems for rendering presentation descriptions in concert.

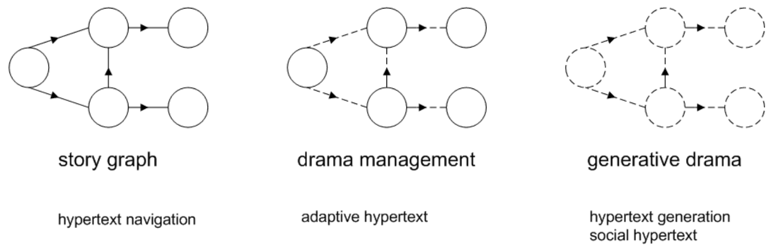


Figure 4.5: Story graphs, drama management and generative drama

Figure 4.5 shows how beat sequencing and run-time authoring can be related to different forms of hypertext. During beat sequencing (drama management) the nodes in the story graph are fixed but the traversal of links is context-dependent and by setting and retracting triggers, links can be enabled or disabled at run-time. In the case of run-time authoring, the nodes are also modified dynamically. This form of authoring can be seen in the form of *social hypertext* on the Web [Erickson, 1996] in many ways (e.g. weblogs, social networking sites). [Güven & Feiner, 2003] is an example of a physical hypermedia system that creates location-based hypermedia narratives before the real-world experience has taken place, [Weal, Michaelides, Thompson & DeRoure, 2003] of modification after the real-world experience has happened. An example of run-time modification of mobile context-aware hypermedia is presented in [Hansen, Bouvin, Christensen & Grønbæk, 2004]. The authors that add and remove beats in an ambient narrative could be (remote) human actors but also software agents as suggested by e.g. [Bleeker, 2006]. The advantage of computer actors is that the author of the ambient narrative does not have to explicitly encode all media-enhanced social scripts manually. It also allows more variation and surprise in the behavior of the intelligent environment. This automatic on-line generation of beats is referred to as *generative drama* by [Mateas & Stern, 2003].

4.5.2 Mass customization of ambient narrative fragments

The key requirement for the ambient narrative model is that the application intelligence can be broken down into modular fragments that are reassembled based on user interaction into a coherent story or ambient intelligent experience. This idea is also used in interactive drama approaches [Laurel, 1986; Bates, Loyall & Reilly, 1991; Perlin & Goldberg, 1996; Magerko, 2002; Mateas & Stern, 2003] and the ambient narrative model has been inspired by this work. [Laurel, 1986] is the first to describe a (hypothetical) drama manager that guides an interactive story experience. This drama manager would sequence story fragments into a coherent narrative based on the choices of the reader in the fantasy world. The Oz project [Bates, Loyall & Reilly, 1991] at CMU and the Improv project [Perlin & Goldberg, 1996] at MIT are early examples of interactive drama systems that implement a drama manager that influences the virtual character's minds, the virtual world in which they live and the user interface through which the user sees the virtual world and is allowed to interact with this virtual world. More recently, [Mateas & Stern, 2003; Szilas, 2003] have demonstrated interactive storytelling systems with a central drama manager that orchestrates the action. The ambient narrative model was inspired by this Façade system in several ways: Mateas and Stern refer to their atomic story fragments as beats and introduce a drama manager that sequences beats from a bag of beats based on explicit and implicit feedback from the story world and a story memory. The selected beat affects the story world and the choices the virtual characters in the story world can take. The actions of the virtual characters in turn influence the beat sequencing process. A human player is also viewed as a character in the story; explicit feedback of the user will cause story values (specified in the beat) to change that will in turn affect the beat sequencing process. Furthermore Mateas and Stern define A Behavioral Language (ABL) that lets authors write parallel, sequential and joint behaviors to create believable agents with (sub)goals and behaviors. An activity (e.g. walking to the player) is represented as a goal, and each goal is supplied with one or more behaviors to accomplish its task. The notion of goals is not present in the ambient narrative model, but goals can be set by using the story memory and introducing beats that test on this value (goal completion). The device action command corresponds with the behaviors in ABL. Like the preconditions in the ambient narrative model, the Beat Sequence Language developed for Façade allows the author to annotate each beat with selection knowledge. The 'init', 'main' and 'final' section in the ambient narrative model are inspired by the possibility in Façade to set actions that need to be performed at various stages in the beat selection process. [Dow, Mehta, Harmon, MacIntyre & Mateas, 2007] describes experiments with an augmented reality version of the Façade system [Mateas & Stern, 2003] but apart from the location of the player in the real world (which is used to influence the story) this version seems identical to the original version except for a head-up display instead of a computer screen. Magerko's IDA architecture [Magerko, 2002] is another example. It represents plots at the scene level and consists of five stages: initial state, required events, background knowledge, content constraints and temporal constraints. The initial state sets up the scene and is similar to the initialization part in our action stage. The required events and background knowledge are comparable with beat preconditions in the ambient narrative model, while the content constraints that limit the binding of the variables used in the required events refer to device action command templates. Like Façade, the IDA architecture has been designed with screen-based interactive storytelling in mind and therefore needs to be extended to support timing and synchronization of device actions and aggregation of sensor data from different sources. Furthermore, these interactive storytelling systems assume an author who has sufficient programming skills to write the parallel behaviors and behavior mixing and therefore do not discuss run-time modification of beats by end-users.

Reassembly of ambient narrative fragments through performance in the real world can be seen as browsing. Triggers (Definition 4.2) in the ambient narrative model correspond with the notion of context-aware links that are implicitly traversed based on user's actions. An example is the HyCon system [Hansen, Bouvin, Christensen & Grønbæk, 2004]. Using the HyConExplorer users can create links, annotations, and trails of links which are automatically tagged with context information sensed from the physical environment. When the user moves about in the physical world, hypermedia structures are presented to the user, if the user's context matches that of the structures: A mobile client captures the contextual information from the physical environment and sends it to a server that returns the relevant hypermedia structures. The trigger precondition and link in the ambient narrative model are similar to the annotation and link in HyCon. [Millard, Roure, Michaelides & Thompson, 2004] explores how various navigational models and link structures can cope with mixed reality scenarios. The trigger precondition and link in the ambient narrative language model corresponds with the source and traversal information (that describes how to get from source to destination) in the FOHM model. During the beat sequencing process the context (Definition 4.4) and story memory (Definition 4.5) determine whether certain context-aware links will be traversed. Our beat sequencing approach can therefore also be compared with the mass customization process taking place in an adaptive hypertext [Brusilovsky, 1996; De Bra, Houben & Wu, 1999] system that guides the navigation through a hypertext graph based on a model of the user and the environment.

Viewing physical performance in ambient intelligence environments as browsing differs from browsing a hypermedia document on an augmented reality or mobile context-aware application display in several ways. Since there are multiple users performing in an ambient narrative at the same time, multiple paths in the graph can be traversed simultaneously by different readers. These reading paths must be able to mutually influence each other as discussed in section 4.3.1. In addition, there is one shared rendering space in ambient intelligent environments as opposed to individual tabs, windows or screens running on different mobile devices. This in turn requires that the rendering engine can dynamically change the set of action scripts it is interpreting/rendering. For more information we refer to section 6.2.3. Dynamic updating support is also need to enable run-time modification of ambient narrative fragments because action scripts of beats that are to be removed must be retracted from the rendering engine to prevent further device action commands specified in those beats.

4.5.3 Social scripts ...

The first part of Requirement AN.1 (Social scripts enhanced with ambient technology) is addressed by the beat preconditions (Definition 4.10) that specify the restrictions on the physical context and ambient narrative state.

Many hypertext models that consider physical context are either targeted towards mobile context-aware applications [Grønbæk, Vestergaard & Ørbæk, 2002] or augmented reality applications [Sinclair, Martinez, Millard & Weal, 2002; Grønbæk, Ørbæk, Kristensen & Eriksen, 2003; Romero & Correia, 2003]. The position and orientation of the user carrying or wearing the device in the physical world alone is often sufficient for these types of applications. For example [Grønbæk, Vestergaard & Ørbæk, 2002] describes a geo-spatial hypermedia system that helps users with a mobile device in the field and people working behind a PC in an office together. The Topos prototype system is intended as an information organization tool that helps an architect on site to bring the electronic documents and models relating to the project with him. [Romero & Correia, 2003] discuss a hypermedia model in a physical interactive storytelling environment and an augmented reality game that takes place in a gallery environment. The player wears a head-up display that superimposes 3D objects that have been selected from a hypermedia graph based on the position of the user in the real world and the state of the narrative.

To describe the social scripts in realistic ambient intelligent environments in which multiple actors can be involved in multiple activities using multiple devices and applications supporting these performances simultaneously, a richer context model is needed. The Contextual Media Integration Language (CMIL) [CMIL, 2008] is intended for annotating multimedia content with contextual information for browsing and information retrieval purposes.

Besides the ability to describe meta-data about multimedia clips (author, keywords, copyright), system and device capabilities (screen-size, modality, language) and network bandwidth (bitrate) the CMIL specification also presents language elements that enable an author to define location (loc), orientation (orient), time and environmental conditions (custom att) for a digital content item. Instead of a single digital content item the ambient narrative model uses an action description (Definition 4.22) to describe what happens when the context preconditions are satisfied. The Augmented World Model (AWL) [Nicklas & Mitschang, 2001] in the Nexus project aimed towards developing an open framework for providers of location-based information. It takes an object-oriented approach to model the real world, augmented by additional information (e.g. web site). The authors additionally define an Augmented World Query Language that supports object retrieval within the AWL model. The AWL model consists of over 120 object classes, most of them static objects (e.g. 'building', 'room', 'road'). A similar approach is taken by [Goβmann & Specht, 2002] that use a world model, augmentation layer and in addition a domain and user model to model augmented environments. Definitions 4.11 to 4.18 were based upon the list of context parameters in Table 3.1 (p.41), inspired by the dramaturgical theory presented in Chapter 2 and supported by the existing hypertext models above. The story value (Definition 4.20) restrictions in the beat preconditions to model tests on the narrative state have been inspired by interactive storytelling systems, in particular [Mateas & Stern, 2003; Szilas, 2003] that also maintain a story memory or session state and allow operations on this state.

In the ambient narrative model, the social scripts could have been represented by RDF [RDF, 2008] or OWL [OWL, 2008] statements and rules about the behavior of people, objects and devices in physical environments. [Ter Horst, Van Doorn, Kravtsova & Ten Kate, 2002; Chen, F. Perich & Chakraborty, 2004] for example use Semantic Web technology to represent context facts and rules for ubiquitous computing environments. The subtree matching process however as discussed in section 4.2 requires pairwise comparison (and calculation) of nodes in the context and preconditions trees instead of logical inference. It is insufficient for example to merely infer whether there is a time element in the context model under a certain stage. We have to calculate whether the context time is within the time restrictions set by the preconditions of the beat or trigger. Although Semantic Web notation could still be used for describing the social script, we opted for simple XML notation.

4.5.4 ... enhanced with ambient technology

The second part of Requirement AN.1 (Social scripts enhanced with ambient technology) is covered by the beat action (Definition 4.21). It specifies the operations on the ambient narrative state before, during and after activation of a beat and the device action commands sent during and after beat activation for rendering. The device action commands (Definition 4.23) are part of an action script (Definition 4.22) that specifies the timing and synchronization of actions over multiple devices (Requirement RT.4).

Originally, we started out using the Synchronized Multimedia Integration Language (SMIL) [SMIL, 2008] as the basis for the action script language. SMIL is inspired by the Amsterdam Hypermedia Model (AHM) [Hardman, Bulterman & Rossum, 1994] which extends the Dexter hypertext reference model [Halasz & Schwartz, 1994] by adding the specification of complex temporal relationships among data items, high-level presentation attributes and link context that are important in supporting hypermedia. Furthermore, the *ref* media object element in the SMIL specification can be used to describe generic media references which may also be used to specify light or other ambient effects and/or specific software applications. SMIL and AHM assume hypermedia presentations running on a single device (central control). A way to extended the SMIL language to include timing and synchronization of multimedia presentations over a set of devices is described by [Doornbos, Clout & Ten Kate, 2003]. Another example of a platform, infrastructure and language to describe and render distributed ambient media is amBX [Eves, Cole & Callaway, 2008]. amBX has been designed for the home domain with unknown configurations of devices in mind. The ambient narrative model however assumes the (relative) location of devices is known and stored in a context model (Definition 4.4) that is checked during a beat sequencing cycle. The AHM and SMIL models are more powerful in expressing hypermedia presentations than the action scripts in the formal ambient narrative model. For example, the layout of multimedia objects on an individual screen cannot be specified. The action script language is however light-weight and easier to implement/modify and covers the basic minimum requirement of timing device actions over multiple devices. In addition, other presentation markup languages and rendering engines are allowed in the action section of a beat by specifying the action type attribute (ac_{ty}) in Definition 4.21.

4.5.5 Run-time modification of ambient narrative fragments

To find a suitable formalism for action selection that meets the requirement of run-time modification of narrative fragments we studied different approaches: The Façade and many other interactive storytelling systems e.g. [Szilas, 2003; Mott & Lester, 2006; Cavazza, Lugrin, Pizzi & Charles, 2007] use AI planning techniques to sequence plot elements into a compelling dramatic story. [Davenport & Murtaugh, 1997] describes another form for action selection using a spreading activation network based on [Maes, 1989] that is used to select relevant story elements from a multimedia database and dynamically join them into an appealing coherent narrative presentation. A Bayesian network approach for computational storytelling in an immersive narrative space is presented by [Sparacino, 2003]. The interval scripts formalism introduced by [Pinhanez, Mase & Bobick, 1997] enable authors of interactive systems to declare time intervals corresponding to the different actions and events and the temporal relationship (e.g. sequence, overlap, meet) between some of those pairs of intervals. This approach is used for a story-based, interactive system named *SingSong*. The Trellis system explained in [Stotts & Furuta, 1989] uses a formal Petri net based model to model synchronization of simultaneous traversals of separate paths through a hypertext. A context-aware version of Trellis (caT) that adapts the presentation of documents based on changes in environmental information such as time, location, and bandwidth/cost is described in [Na & Furuta, 2001]. When the reader browses caT documents, the system provides dynamic documents from its collection, incorporating the reader's contextual and preference information.

The downside of these formalisms is that they assume a closed and fixed set of fragments whereas the run-time authoring requirement states that humans and/or software agents should be able to add or modify story fragments while the story is progressing. For example, Petri nets are typically used to model closed environments, and the probability distributions in a Bayesian network must be updated each time a new node is added or an existing one removed. For this reason we decided to implement the ambient narrative as described in section 4.4.

4.6 Concluding remarks

In this chapter we presented the formal ambient narrative model and compared this model with existing work in (adaptive) hypermedia, interactive drama and mixed reality. An ambient narrative is seen as a hypertext graph in which the nodes (beats) represented situated actions and the links (triggers) transitions between nodes. By performing in the physical environment, people, objects and devices implicitly activate triggers and traverse this hypertext graph. The collective performance is the combination of these (interacting) hypertext readings. In contrast to mobile context-aware or augmented reality applications, there is only one (shared) rendering space extends across multiple devices which implies that the rendering engine must support dynamic updating of action scripts and timing and synchronization across devices.

Furthermore, we presented how the hypertext structure can be updated at run-time to support dynamically changing ambient narratives. Because each narrative fragment both reads (through testing its preconditions on the context and session state) and writes (by sending action scripts for rendering and modifying the story memory and/or list of active triggers) information, the difference between reading and writing text, consuming and producing symbolic meaning, is only visible at the lowest level in the model. This offers the possibility to build more complex dynamically changing ambient environments on top of this formal model when we consider beats as 'atoms' and ambient narratives as 'molecules' that are subjected to (non-)deterministic 'forcefields' (changes in the context and session state) and other beat atoms and ambient narrative molecules to which they react.

In the next chapter we discuss the implementation of (a subset of) the ambient narrative model in the ambient narrative engine that forms the central component in the system architecture that will be discussed in Chapter 6.

5

Ambient Narrative Engine

After having elicited the requirements in Chapter 3, we have formalized these requirements and presented a formal approach in Chapter 4. In this chapter we further specify and describe the realization of the *Ambient Narrative Engine* component that implements the formal ambient narrative model and core language subset of Table 4.1. This component forms the central component of the shop window system in Chapter 6.

The outline of this chapter is as follows: Section 5.1 describes the storage and retrieval strategy for representing and managing the state of an ambient narrative. The beat sequencing algorithm is discussed in Section 5.2. Section 5.3 introduces and explains the algorithm to modify the beat set at runtime. This chapter concludes by mapping these algorithms to classes in the architecture of the *Ambient Narrative Engine* component that implement this functionality (Section 5.4).

5.1 A database approach

To maintain the state of an ambient narrative over time persistent storage is needed. In Section 4.1 the state of an ambient narrative (Definition 4.26) was defined by the tuple (Ctx, Mem, B, AB, T) where Ctx represents the context situation (Definition 4.21), Mem the session state in the story memory (4.22), B

the beats in the beat set (4.23), *AB* the active beat set (4.25) and *T* the trigger list (4.24). One possible approach is to store the beats in the beat set and context situation as files on disk and load these files at run-time. This thesis proposes an alternative approach where the beat sequencing problem is seen as a retrieval problem where the physical context, session state and the trigger set represent the user query and the (active) beat set the data set. By storing the state of the ambient narrative in a modern database management system (DBMS) *physical* and *logical data independence* [Tsichritzis & Klug, 1978] is offered to users: Physical data independence hides the physical storage structure and access (file allocation, indexing, cache optimization) for the user, so the author of the ambient narrative does not need to worry about which storage strategy is most efficient given the size and type of queries performed on the beat set. Logical data independence allows for changes in the logical schema without having to rewrite existing applications. For example, if an author uses a relative XPath location path in a trigger link to select all beats that have a precondition element with an actor descendant node with a particular id, the trigger link does not need to be updated if the beat precondition schema is modified (unless the actor node is no longer a descendant of a 'pre' node). In addition, complex queries, instead of links, can be formulated that select multiple beats at once and/or aggregate data from different beats in new beats. Finally, if multiple applications want to use the same data as the beat sequencing application, compatibility is less of an issue for the application developer.

Because an ambient narrative can be seen as a graph consisting of nodes (beats) and edges between these nodes (triggers) with hierarchical beat nodes, we can represent them as XML documents as discussed in Section 4.1. To store and retrieve these XML documents we use *eXist*, an open-source native XML DBMS [EXist, 2008] (version 1.1). This DBMS implements the XPath [XPath, 2008] and XQuery [XQuery, 2008] standards for querying XML documents and the XUpdate [XUpdate, 2008] specification for updating XML documents[1].

In a first version of the ambient narrative engine that implemented the beat sequencing algorithm, the complete state of an ambient narrative was stored in XML documents and maintained by the *eXist* DBMS. The precondition checking, beat sequencing and run-time authoring operations were written down as XQuery/XUpdate commands that would retrieve, manipulate and write back the new state in the database. This approach is closest to viewing

[1]The W3C XQuery Update Facility [XQueryUF, 2008] standard was not yet available at the time of writing the Ambient Narrative Engine component

the beat sequencing problem as an information retrieval problem and places the entire state of an ambient narrative engine under control of the DBMS. This implementation however did not satisfy Requirement RT.5 (*No noticeable difference in performance with a custom-built system*): The execution of the more complex XQueries as the precondition checking expression in Appendix A by the *eXist* engine took in the order of hundreds of milliseconds and this led to unacceptable long beat sequencing cycles and a slow response in the intelligent shop window environment.

For the final version this complete mapping was abandoned in favour of a mapping where only the context and beat set are stored as XML documents. The context, story memory and beats in the beat set are automatically loaded into objects in memory at start-up or when first accessed (in case of beats). Modifications on the context XML document and beat XML documents are implemented by XUpdate expressions. In case of such a modification, the memory object must be updated to ensure consistency. Adding and removing beat XML documents is done using an eXist-specific API. For finding the target of a link and querying the beat set, XPath expressions on the beat XML documents are used. More detail on their usage will be given in the discussion of the beat sequencing and run-time algorithms that follow next.

5.2 Beat sequencing

In Section 4.3 the beat sequencing process was defined as the problem of determining the new state of an ambient narrative ($an \rightarrow an'$) after a change in context ($Ctx \rightarrow Ctx'$) or change in session state ($Mem \rightarrow Mem'$). This section describes how this beat sequencing process is implemented by the *Ambient Narrative Engine* component.

Algorithm 1 lists the pseudocode to update the state of an ambient narrative after a context or session state change event. A completed state update is one beat sequence cycle. The algorithms of procedures specified in small capitals can be found in the text of this chapter. The BEATSEQUENCING algorithm starts if an event is received that represents a context change or story value change.

First, a lock is created on both the context model and story memory to ensure that all reads and updates during a transaction (beat sequence cycle) operate on the same data. The *snapshot isolation* of the context model ensures that one thread can update the context model while another thread can update the state of the ambient narrative. Because a beat sequencing cycle does not modify the context model, there is no need to check for conflicts once the transaction has finished. Concurrent transactions on the story memory are not

Algorithm 1: BEAT SEQUENCING

```
Input: $event$
Output: $AB$, $B \setminus AB$
LockState();
if (UPDATECONTEXT($event$) $\wedge$ UPDATESTORYMEMORY($event$) == 0) then
    UnLockState();
    return
end
Initialize set of all invalid beats: $IS$ = GETINVALIDBEATS($AB$);
Initialize set of all activated beats: $AS = \emptyset$;
if ($|IS| > 0$) then
    UPDATEACTIVEBEATSET($AS$,$IS$);
end
RECOMPUTETRIGGERS($AS$,$IS$);
RemoveDuplicates($AS$,$IS$);
SENDACTIONSCRIPTS($AS$,$IS$);
NotifyListeners($AB$,$B \setminus AB$);
UnLockState();
```

allowed, only the beat sequence thread can modify the story memory. This way there is also no need to check for conflicts on the story memory. The lock on the context model and story memory must be released if the beat sequence cycle has been completed.

If the UPDATECONTEXT procedure changes the context model or if the UPDATESTORYMEMORY procedure alters the state of the story memory, the new state must be calculated. This proceeds in a number of steps. First, the set of invalid beats is calculated by the GETINVALIDBEATS procedure. If the preconditions of a beat in the active beat set are no longer satisfied by the new context and memory state, the active beat set must be modified. This is done by the UPDATEACTIVEBEATSET procedure. If the UPDATEACTIVEBEATSET procedure has completed the resulting set of activated beats and set of invalid beats is used as input for the RECOMPUTETRIGGERS procedure that checks for triggers that have been activated by the state change.

When the RECOMPUTETRIGGERS procedure has finished, the lists of activated and invalid beats need to be compared and checked for duplicate entries. In one beat sequence cycle a beat can be added and removed again. These pairs can be filtered out because they do not cause an effect on the outside world in terms of changes in the active beat set or device actions that are rendered in the environment. For example, the activated beat set $AS = B, A, C$ and invalid beat set $IS = A, D, B$ result in $AS = C$ and $IS = D$ after removal of duplicate, temporary beat entries.

After the activated beat set and invalid beat set have been filtered, the action section of these beats is processed by the SENDACTIONSCRIPTS procedure for action scripts that need to be rendered by the rendering engine. All clients interested in changes of the active beat set and beats in the beat set that are currently inactive are notified (NotifyListeners).

5.2.1 Updating the logical context model

The UPDATECONTEXT procedure described by Algorithm 2 expects a context event with an operation type, element name, parent object and a list of attribute name, value pairs. It parses the event and generates an XUpdate query that is executed on the logical context database. The UPDATECONTEXT procedure essentially creates an XML node that is either inserted in or removed from the context XML document tree depending on the operation type.

Algorithm 2: Update Context

```
Input: event
Output: 1 if the context event has been processed, 0 otherwise
query = CreateContextQuery(parseEvent(event));
updated = Update(contextDB,query);
if updated then
    return 1 ;
end
else return 0 ;
```

The *Ambient Narrative Engine* and UPDATECONTEXT procedure operate on a *logical* context model, i.e. a symbolic description of the physical environment that abstracts away from the exact physical coordinates of areas, people, and things and time. The *physical* context model that maintains a view of the raw sensor data is maintained by the *Context Server* component that will be discussed in Section 6.2.1. The reasons for separating the context model in a physical context model and a logical context model are scalability and portability. If all raw sensor data would be fed through a central reasoning component this would create a potential performance bottleneck in the system. By working on a symbolic description of the physical environment that records only the relevant entities and their relations, the semantics of the original physical environment remains intact but data with little or no meaning is filtered out. This reduces the workload on the beat sequencing process and enables the ambient narrative engine to reason about information coming from many sensors. A second benefit of distinguishing between a physical and logical context model is that the geometric and site specific information is defined in the physical context model. Only this model needs to be copied and modified to port an ambient narrative from one place to another. This however does not imply the physical context information should be hidden away from the application layer as some applications started by device action command may need to have access to real-time data streams from one or more sensors. The ambient narrative *Editor* component (Section 6.4) needs access to the physical context information to update the position of people and objects in the shop window area, for example. Similar to [Ranganathan, Al-Muhtadi & Chetan, 2004] we offer applications therefore access to both the physical and logical context model.

To explain how a logical context event is processed by the UPDATECONTEXT procedure consider the following example. Suppose an actor named *actor1* in role *customer* appears on a stage called *interact1*. This logical context event is represented by the following event message:

```
add context interact1 actor id actor1 role customer
```

After parsing this event, the following XUpdate query is created and executed on the context database, adding an actor node in the context tree under a performance node that has a stage element parent node with an attribute id having the value *interact1*:

```
<xupdate:modifications version="1.0"
xmlns:xupdate="http://www.xmldb.org/xupdate">
<xupdate:remove
   select="/context/pre/stage[@id='interact1']/*/actor[@id='actor1']/>
<xupdate:append
   select="/context/pre/stage[@id='interact1']/performance"
   child="last()">
<actor id='actor1' role='customer'/>
</xupdate:append>
</xupdate:modifications>
```

Note that an actor can be in multiple symbolic locations (stages) simultaneously (as stages can overlap) but an actor can only be present on each stage once.

5.2.2 Updating the story memory

Algorithm 3 shows the UpdateStoryMemory procedure which implements the operations 4.3, 4.3 and 4.3. If a story value event has been received, the story memory is updated depending on the story value action and whether a story value with the same id already exists in the story memory in which case the previous value and new value need to be added. Updating an existing story value supports the construction of *story arcs* as used in Façade system [Mateas & Stern, 2003]: Beats can cooperate to increase or lower a story value until a beat becomes active that retracts the story value.

Algorithm 3: UPDATE STORY MEMORY

```
Input: $sv$
Output: 1 if the story value event has been processed, 0 otherwise
if ($sv_{ac}$ == 'add') then
    if ($Mem$ contains $sv$) then
        $m$ = Get($Mem$,$sv$);
        Replace($Mem$,$m$,$m_{val} + sv_{val}$);
    end
    else
        $Mem \cup sv$;
    end
    return 1;
end
if ($sv_{ac}$ == 'remove') then
    $Mem \setminus sv$;
    return 1;
end
return 0;
```

5.2.3 Determining invalid beats

The implementation of the GETINVALIDBEATS procedure is described by Algorithm 4 which fulfills Equation 4.3.3. For each beat in the active beat set we need to check if its preconditions still hold against the new updated context model and story memory by calling the CHECKPRECONDITIONS procedure. If this procedure returns false the beat is added to the invalid beat set.

Algorithm 4: GET INVALID BEATS

```
Input: active beat set $AB$
Output: invalidated beats $IB$
beat $b$;
$IB = \emptyset$;
foreach ($b \in AB$) do
    $res$ = CHECKPRECONDITIONS($b/pre, Ctx, Mem$);
    if ($res$ == 0) then $IB \cup b$
end
return $IB$;
```

5.2.4 Checking beat and trigger preconditions

The precondition checking algorithm is used to determine if the precondition section of a beat or trigger have been satisfied or not (Algorithm 5). The CHECKPRECONDITIONS procedure is used by the GETINVALIDBEATS procedure to determine if active beats are no longer valid and by the RECOMPUTETRIGGERS procedure if triggers in the trigger set need to be activated and their links/queries executed.

Algorithm 5: CHECK PRECONDITIONS

```
Input: Precondition $Pre$, context $Ctx$ and story memory $Mem$
Output: 1 if the beat precondition is satisfied, 0 otherwise
$stageValid, performanceValid, aCount, prCount, svCount = 0$;
Initialize performance table: $P = \emptyset$;
$s^p = Pre/stage$;
foreach $(Stage\ s^c\ in\ Ctx/stage)$ do
    $stageValid$ = StageCheck($s^p$,$s^c$);
    if $(stageValid == 1)$ then break;
end
if $(stageValid == 0)$ then return 0;
foreach $(Performance\ p^p\ in\ s^p/performance)$ do
    foreach $(Performance\ p^c\ in\ s^c/performance)$ do
        $performanceValid$ = PerformanceCheck($p^p$,$p^c$);
        if $(performanceValid == 1)$ then
            $P \cup (p^p,p^c)$;
            $pCount++$;
            break;
        end
    end
end
if $(pCount \neq |\ s^p/performance\ |)$ then return 0;
foreach $Performance\ p^p$ in precondition performance list of $P$ do
    $p^c$ = Context performance restriction for $p^p$ in $P$;
    if $(ActorCheck(p^p,p^c) == 1)$ then $aCount++$;
    if $(PropCheck(p^p,p^c) == 1)$ then $prCount++$;
end
if $(aCount \neq |\ P\ |) \vee (prCount \neq |\ P\ |)$ then return 0;
foreach $(Story\ value\ sv^p\ in\ Pre/storyvalue)$ do
    foreach $(Story\ value\ sv^c\ in\ Mem)$ do
        if $(StoryValueCheck(sv^p,sv^c) == 1)$ then $svCount++$;
    end
end
if $(svCount \neq |\ Pre/storyvalue\ |)$ then return 0;
return 1;
```

The CHECKPRECONDITIONS procedure provides a partial implementation of the formal precondition checking model of of Section 4.2. More precisely, it implements the necessary precondition tests to support the core language subset of the ambient narrative model in Table 4.1 with the exception of the position test that is performed outside the ambient narrative engine component.

The precondition checking algorithm assumes each precondition has only one stage with each stage having one or more performances. The StageCheck, ActorCheck, PropCheck and StoryValueCheck procedures implement the formal precondition tests defined by Equations 4.2.1, 4.2.5, 4.2.6 and 4.2.9, respectively. For the intelligent shop window scenario *presence* is the only performance that is detected and reasoned about so the PerformanceCheck procedure (Equation 4.2.4) tests only on whether there is a presence performance or not. The time, condition, orientation precondition tests are not implemented (Equations 4.2.2, 4.2.3 and 4.2.7). The position test (Equation 4.2.8) is handled outside the ambient narrative engine and partially implemented; only tests on whether an actor or prop is within a certain stage marked by absolute coordinates are possible (*Co* attribute only in Definition 4.9). A prop can also be set to be omnipresent to make it part of every stage. To test whether an actor or prop is on a certain stage, the coordinates of the corners of the stage and the position of the actor or prop are needed. This information is part of the physical context model and is maintained by the *Context Server* component that will be discussed in Section 6.2.1.

The order of the precondition tests can be varied to try out different optimization strategies. Algorithm 5 uses a breath-first search strategy that starts by searching for a stage node in the context tree that matches the stage precondition node. If the context stage node does not pass the stage test, the node and its child nodes are pruned from the search tree.

The difference in implementation of the core language subset and the complete specified language model with regard to the Ambient Narrative Engine is in the CHECKPRECONDITIONS and UPDATECONTEXT procedures. To implement the complete language model, the CHECKPRECONDITIONS procedure needs to implement additional precondition tests and the UPDATECONTEXT procedure must be extended to be capable of inserting or removing such nodes in the logical context model. The main reasons for not implementing this complete language model here is not so much the complexity of the ambient narrative engine but the shop window carrier application and the *Context Server* and *Editor* components in the ambient narrative system architecture that are discussed in the next chapter.

5.2.5 Updating the active beat set

The UPDATEACTIVEBEATSET procedure described by Algorithm 6 manages the active beat set and looks for trigger and story value elements in the final section of beats that are removed from the active beat set and in the init section of beats that need to be added to the active set. Triggers and story values in the main section are not allowed. The UPDATEACTIVEBEATSET procedure implements the operations ADDACTIVEBEAT and REMOVEACTIVEBEAT (section 4.3) but does not send the action scripts to the rendering engine. This is done at the end of the beat sequence cycle after the RemoveDuplicates procedure to prevent temporary beats from influencing the rendering process.

Algorithm 6: UPDATE ACTIVE BEAT SET

```
Input: $activatedBeats$, $invalidBeats$
Output: $AS$, $IS$
foreach ($b \in invalidBeats$) do
    $FT = b//final/trigger$;
    foreach ($t \in FT$) do
        UPDATETRIGGERLIST($t$,$T$);
    end
    $FSV = b//final/storyvalue$;
    foreach ($sv \in FSV$) do
        UPDATESTORYMEMORY($sv$,$Mem$);
        if (Mem has changed) then
            $moreInvalidBeats$ = GETINVALIDBEATS($AB$);
            UPDATEACTIVEBEATSET($\emptyset$,$moreInvalidBeats$);
        end
    end
    $AB \setminus b$;
    $T_{AB} \setminus t_b$;
end
foreach ($b \in activatedBeats$) do
    $AB \cup b$;
    $T_{AB} \cup t_b$;
    $IT = b//init/trigger$;
    foreach ($t \in IT$) do
        UPDATETRIGGERLIST($t$,$T$);
    end
    $ISV = b//init/storyvalue$;
    foreach ($sv \in ISV$) do
        UPDATESTORYMEMORY($sv$,$Mem$);
        if (Mem has changed) then
            $moreInvalidBeats$ = GETINVALIDBEATS($AB$);
            UPDATEACTIVEBEATSET($\emptyset$,$moreInvalidBeats$);
        end
    end
end
$AS \cup activatedBeats$;
$IS \cup invalidBeats$;
```

The UPDATEACTIVEBEATSET procedure expects two lists, a list of beats that must be added to the active beat set (activated beats) and a list of beats that must be removed from the active beat set (invalid beats).

First the final section of each beat in the invalid beat list is checked for trigger and story value commands. If a trigger command has been found, the UPDATETRIGGERLIST procedure is invoked to update the trigger set. If a story value command is encountered, the story memory is updated by calling the UPDATESTORYMEMORY procedure. After these changes have been committed, the beat can be removed from the active beat set and the corresponding trigger in the active trigger list can be removed.

For each beat in the activated beat list we first add the beat to the active beat set and its corresponding trigger to the active trigger list. Next, the ‘init’ section of each beat in the set of activated beats is inspected for trigger and story value commands. The trigger list and story memory are updated if trigger and/or story value commands have been encountered.

Note that after the story memory has changed, new beats may have become invalid. The UPDATEACTIVEBEATSET procedure must be invoked with this new set of invalid beats until the set of invalid beats remains unchanged.

5.2.6 Updating the trigger list

Algorithm 7 describes the implementation of the UPDATETRIGGERLIST procedure that fulfills operation 4.3.

Algorithm 7: UPDATE TRIGGER LIST

```
Input: trigger $t$, trigger list $T$
Output:
if ($t \notin T$) then
    if ($t_{be}$ == 'add') then
        $T \cup t$;
    end
    if ($t_{be}$ == 'remove') then
        $T \setminus t$;
    end
end
```

5.2.7 Testing on new triggered links

After a context or story memory change the BEATSEQUENCE procedure must call the RECOMPUTETRIGGERS procedure (Algorithm 8) to check for newly activated triggers.

For each trigger in the trigger list that is not yet active, the trigger preconditions need to be checked (Equation 4.3.4). If the preconditions match, the trigger link is added to set of activated links.

For each link in the activated link set the *to* and *from* link attributes need to be processed. The value of the *to* attribute is an XQuery expression that must be executed on the beat set, the value of the *from* attribute is an XQuery expression that is executed on the active beat set. Although complex XQuery expressions are allowed, the shop window system uses simple XPath expressions to select individual beats for addition or removal. For example the XPath ex-

Algorithm 8: RECOMPUTE TRIGGERS

```
Input: $AS$, $IS$
Output: $AS$, $IS$
Initialize activated links: $activatedLinks = \emptyset$;
Initialize activated beat set: $activeBeats = \emptyset$;
Initialize invalidated beat set: $invalidBeats = \emptyset$;
$res = 0$;
foreach ($t \in T; t \notin T_{AB}$) do
    $res$ = CHECKPRECONDITIONS($t/pre$,$Ctx$,$Mem$);
    if ($res == 1$) then $T_{AB} \cup t$;
    $activatedLinks \cup t/link$;
end
foreach ($l \in activatedLinks$) do
    if $l_{to}$ then $activeBeats \cup$ ExecuteQuery($B$,$l_{to}$);
    if $l_{fr}$ then $invalidBeats \cup$ ExecuteQuery($AB$,$l_{fr}$);
end
if $(|activeBeats| = 0) \wedge (|invalidBeats| = 0)$ then
    return $AS$, $IS$;
end
$AS$, $IS$ = UPDATEACTIVEBEATSET($activeBeats$,$invalidBeats$);
if $(|AS| > 0) \vee (|IS| > 0)$ then
    RECOMPUTETRIGGERS($AS$,$IS$);
end
```

pression `//beat[@id='attractorMode']` returns a single beat named *attractorMode*.

If beats have been retrieved as a result of these queries, the UPDATEACTIVEBEATSET procedure is called with as parameteres the results of the addition queries and the results of the removal queries. The RECOMPUTETRIGGERS is then called again until no new triggers are activated.

The RECOMPUTETRIGGERS and UPDATEACTIVEBEATSET procedures form the core of the beat sequencing algorithm. In the current implementation first all story value changes are processed and beats are removed until no beats with invalid preconditions exist in the active beat set and the story memory remains unchanged. Then we check for newly activated triggers and also start to add beats to the active beat set. During this process the story memory may change again and newly added beats can be rendered invalid. The process stops if the preconditions of all active beats are still valid (Equation 4.3.1) and no new triggers are activated (Equation 4.3.2).

5.2.8 Sending action scripts

At the end of a beat sequence cycle, after having filtered out temporary duplicate beats the SENDACTIONSCRIPT procedure is called to inform the rendering engine of the changes in the active beat set. Algorithm 9 shows the implementation of this procedure.

Algorithm 9: SEND ACTION SCRIPTS

```
Input: $AS$,$IS$
Output:
foreach ($b \in IS$) do
    $target = b/action/@ta$;
    $type = b/action/@ty$;
    SendData($type$,$target$,$b//final/script$);
end
foreach ($b \in AS$) do
    $target = b/action/@ta$;
    $type = b/action/@ty$;
    SendData($type$,$target$,$b//main/script$);
end
```

For each beat in the invalid and active set the rendering engine machine address and type is looked up by reading the *target* and *type* attributes of the action element of the beat. This information is used to send the main (for beats in the active set) or final (for beats in the invalid set) action script containing device action commands to the right rendering engine.

5.2.9 Beat sequence cycle control

The RECOMPUTETRIGGERS and UPDATEACTIVEBEATSET algorithms can lead to long or infinite beat sequence cycles because of their recursive nature. Inside a beat sequence cycle the context itself is never modified and since beats cannot modify the context model, the problem can be traced back to the story value mechanism. For example if beat A tests on a story value X=1 in its precondition part and sets Y=1 and Z=0 in its action part and a beat B tests on Y=1 and sets X=0 and Z=1 and a beat C tests on Z=1 and sets X=1 and Y=0 the *UpdateActiveBeatSet* procedure will never terminate once beat A is activated and there exists a trigger in the trigger list (or is set by A or B) with a link that points to beat B and C. Contrary to social script conflicts (e.g. one beat sets a light on, the other one switches it off, both under the same set of preconditions), conflicts like these inside a single beat sequence cycle cannot be resolved by the editor but need to be prevented by the engine at run-time in the hypertext graph as it is being constructed and modified.

Section 4.3 discussed rules to ensure the order of story memory and trigger modifications does not affect the outcome. Rule 4.3.7 also prevents the types of conflicts above: If beat B is not allowed to change story value Z to 1, beat C cannot set the story values X and Y. Determining whether this rule is violated can be difficult because beats A and B may modify the trigger list, leading to new beats being added and others removed that have to be checked as well. This process can take place at editing/authoring time when end-users modify the ambient narrative using the authoring environment. Finally we note that this rule can be relaxed in certain situations: If beat B removes the trigger that activates C in the example, the cycle terminates even though Rule 4.3.7 is violated.

The heuristics mentioned in Section 4.3.1 for dealing with race conditions also apply here. General beats (few preconditions) are more likely to cause interference with other beats than specific ones. Keeping the size of the story memory and trigger list small reduces the chance of write conflicts. Finally, the question can be raised how important this problem is in practice. The user-generated ambient narratives of Chapter 2 for example do not (yet?) show this level of complexity. In the next section our policy will be discussed in more detail when we look closer at the types of hypertext graphs end-users are able to make.

5.3 Run-time authoring

The run-time modification of ambient intelligence fragments to support co-creation environments was defined in Section 4.4 as the problem of determining the next state of the ambient narrative, $an \rightarrow an'$, given a change in the beat set, $B \rightarrow B'$. In this section the implementation of the run-time authoring support is discussed in detail. Three different situations can be identified: beat addition, beat removal and beat modification. The implementation of these operations will be explained in detail but first we discuss how the hypertext graph is modified when a beat is added or removed.

5.3.1 Modifying the hypertext graph

By adding and removing triggers in the ‘init’ and ‘final’ section of beats the author of the ambient narrative can create a variety of hypertext graphs. This gives the author control over which beats can be activated in which situation. The author may decide for example to write one beat that sets the triggers for all the other beats in the ambient narrative, create a beat that will remove an number of triggers to prevent a subset of the beats to become active or create more complex schemes where multiple beats contain commands to add and remove triggers.

Essentially, the mechanism to set and retract triggers enables the author of the ambient narrative to write an optimization strategy to manage the size of the trigger list that is accessed by the RECOMPUTETRIGGERS procedure. The advantage of this approach is that the author can implement different optimization schemes without having to change the source code of the ambient narrative engine itself, as this optimization strategy is defined in the ambient narrative itself.

However, understanding and choosing how to distribute set and retract trigger commands over the beats in the ambient narrative can be difficult for experts let alone end-users without a programming background. To support these end-users in creating intelligent shop window applications in our case, we need to choose an optimization strategy in advance that is used to modify the hypertext graph when an end-user adds or remove a beat so that end-users can concentrate on specifying the beat precondition and action.

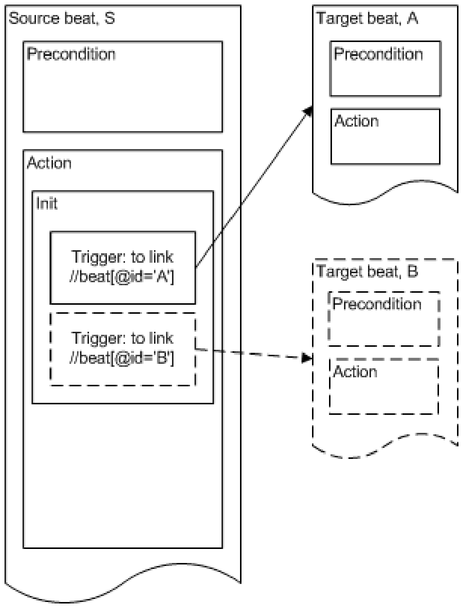


Figure 5.1: Adding a beat node in the hypertext graph

Because an intelligent shop window ambient narrative typically covers only a small area of a store and consists of a small number of beats in total, it was decided to create one start-up beat with empty preconditions so that it would always be active. Figure 5.1 graphically illustrates this strategy. When a user adds a new beat (target beat), a trigger element with the preconditions of this new beat and an add/to link to this new beat is created and appended to the init section of the start-up beat (source beat). When a user removes a target beat its corresponding trigger in the source beat is removed. To update the source beat an XUpdate expressions is generated that appends a trigger to the target beat in the source beat. Adding a target beat is done by adding the XML document to the beat collection, removing by deleting the XML document from the beat collection. This flat structure guarantees that all triggers are always checked by the RECOMPUTETRIGGERS procedure. As the size of the ambient narrative is small this is not an issue. This strategy also prevents non-terminating beat sequence cycles as a result of recursive modifications to the trigger list. The restriction on the story memory modification is built into the ambient narrative editor that will be discussed in Section 6.4. This editor restrains authors from setting story values that have the same id. This works as follows: Once a beat is created by the editor, a story value element with the id of the new beat is added to the init and final sections of the new beat. If the beat is activated, the story value is added, if it is deactivated, the story value is removed. The user interface of the editor enables users to test on these story values and this way create dependencies between different beats.

The intelligent shop window application of Section 3.3 in total needs one start-up beat with triggers to 9 beats; one beat for the product slideshow presentation on the four shop windows, four beats for the interactive shop window presentation on the individual shop windows and another four beats to describe the situation when no person is standing in front of a particular shop window display but there is a person standing in front of one of the other shop window displays.

5.3.2 Adding beats

Algorithm 10 describes how a beat is added to an ambient narrative and how this action affects the beat sequencing process. The ADDBEAT procedure implements operation 4.4.

The ADDBEAT procedure starts by checking if the target beat that must be added already exists in the beat set. If this is the case the target beat is first removed by calling the REMOVEBEAT procedure. After this step the target beat can be stored in the beat collection (StoreBeatOnDisk). Next, the source beat must be updated (UPDATESOURCEBEAT) and the generated XUpdate query applied on the database (ExecuteQuery). Note that for offline authoring we could stop after this step: If the ambient narrative engine is restarted the source beat will be loaded from disk and the beat sequencing process proceeds normally.

The ambient narrative is now updated in the database but the source beat has been altered so we need to reload the source beat to ensure the newly added trigger in the init section is part of the source beat object in memory (ReloadSourceBeat, ReplaceBeat). If the source beat is also present in the active beat set, the old source beat must be replaced by this new source beat (ReplaceBeat). Next, the newly added trigger must be added to the trigger list (UPDATETRIGGERLIST).

Algorithm 10: ADD BEAT

```
Input: Beat $b_t$
Output:
LockState();
if ($b_t \in B$) then
    REMOVEBEAT($b_t$,$B$);
end
StoreBeatOnDisk($b_t$);
ExecuteQuery(UPDATESOURCEBEAT($b_t$,ADD),$beatDB$);
$b_s$ = ReloadSourceBeat($b_t$,$B$);
ReplaceBeat($B$,$b_s$);
if ($b_s \in AB$) then
    ReplaceBeat($AB$,$b_s$);
end
foreach ($t \in b_s//init/trigger$) do
    UPDATETRIGGERLIST($t$,$T$);
end
$AS = \emptyset$;
$IS = \emptyset$;
RECOMPUTETRIGGERS($AS$,$IS$);
RemoveDuplicates($AS$,$IS$);
SENDACTIONSCRIPTS($AS$,$IS$);
NotifyListeners($AB$,$B \setminus AB$);
UnLockState();
```

The RECOMPUTETRIGGERS procedure must be invoked to check if the newly added trigger is fired. The beat sequencing process continues by filtering out the temporary duplicate beats (RemoveDuplicates), sending out the action scripts (SENDACTIONSCRIPTS) and notifying clients interested in changes of the (active) beat set (NotifyListeners).

5.3.3 Updating the source beat

The UPDATESOURCEBEAT procedure is specified by Algorithm 11. After finding the source beat of the target beat (FindSourceBeat), the trigger that points to the target beat must be added to the source beat. The CreateSource-BeatQuery procedure generates the XUpdate query to modify the source beat in the beat set. This query is then executed on the beat set (Update). The source beat is returned if this operation is successful.

To add a trigger refering to a target beat *OnInteractMode1* in a source beat named *OnStartUp*, the following XUpdate expression (precondition markup omitted) is generated and executed on the beat collection:

```
<xupdate:modifications version="1.0"
xmlns:xupdate="http://www.xmldb.org/xupdate"
<xupdate:remove
   select="//beat[@id='OnStartUp']/action//
          trigger[@id=t_OnInteractMode1']"
<xupdate:append
   select="//beat[@id='OnStartUp']/action/init" child="last()">
<trigger id="t_OnInteractMode1" behavior="add">
<pre>...</pre>
<link behavior="add" id="l_OnInteractMode1"
   to="//beat[@id='OnInteractMode1']" />
</trigger>
</xupdate:append>
</xupdate:modifications>
```

Algorithm 11: Update Source Beat

```
Input: beat $b_t$
Output: beat $b_s$
$b_s$ = FindSourceBeat($b_t$);
$query$ = CreateSourceBeatQuery($b_s$,$b_t$));
$updated$ = Update(beatDB,$query$);
if $updated$ then
  |  return $b_s$ ;
end
else return 0 ;
```

5.3.4 Removing beats

The procedure to a remove beat is depicted by Algorithm 12. REMOVEBEAT starts by finding the trigger that refers to the target beat (FindTrigger) that must be removed and removing this trigger from the trigger list (UPDATETRIGGERLIST) to prevent this beat from being activated in the future. If the target beat is in the active best set, its trigger must be removed from the list of activated triggers. A beat sequence cycle is then started with the target beat in the invalid beat set. After the beat sequence cycle has stopped and clients interested in changes of the (active) beat set have been notified (NotifyListeners), we can safely update the beat database. Note that for offline authoring we can start at this point in the algorithm.

First, the source beat must be updated (UPDATESOURCEBEAT) and the generated XUpdate query must be applied on the database (ExecuteQuery). Since the source beat has been altered we need to reload the source beat to ensure the removed trigger is also no longer part of the source beat object in memory (ReloadSourceBeat, ReplaceBeat). If the source beat is also present in the active beat set, the old source beat must be replaced by this new source beat (ReplaceBeat). The target beat can now be deleted in the beat database by calling the RemoveBeatOnDisk procedure.

Algorithm 12: REMOVE BEAT

```
Input: Beat $b_t$
Output:
LockState();
$t$ = FindTrigger($b_t, T$);
UPDATETRIGGERLIST($t, T$);
if ($b_t \in AB$) then
    UPDATETRIGGERLIST($t, T_{AB}$);
    $activatedBeats = \emptyset$;
    $invalidBeats = \emptyset \cup b_t$;
    $AS$, $IS$ = UPDATEACTIVEBEATSET($activatedBeats$,$invalidBeats$);
    RECOMPUTETRIGGERS($AS$,$IS$);
    RemoveDuplicates($AS$,$IS$);
    SENDACTIONSCRIPTS($AS$,$IS$);
    NotifyListeners($AB$,$B \setminus AB$);
end
ExecuteQuery(UPDATESOURCEBEAT($b_t$,REMOVE),$beatDB$);
$b_s$ = ReloadSourceBeat($b_t, B$) ReplaceBeat($B, b_s$);
if ($b_s \in AB$) then
    ReplaceBeat($AB, b_s$);
end
RemoveBeatOnDisk($b_t$);
UnLockState();
```

5.3.5 Modifying beats

To edit a beat b first a copy b' is made that has the same beat id. The original beat is first removed and then the modified copy is added as shown by Algorithm 13 (EDITBEAT).

Algorithm 13: EDIT BEAT

```
Input: Beat $b$, $b'$
Output:
RemoveBeat($b$,$B$);
AddBeat($b'$,$B$);
```

5.4 Component implementation

To conclude this chapter we describe the architecture of the *Ambient Narrative Engine* component and the mapping of the procedures introduced in the previous sections to the classes of this component. This section concentrates on the core functionality, in Section 6.2.2 the interfaces to the other components in the shop window system architecture are discussed. Figure 5.2 shows the UML static structure diagram [Fowler, 2003] of this component in detail. Method and field names have been omitted for readability, only classes and their relations are depicted.

The *Main* class starts the *Engine*. The *Engine* creates two servers to which clients can connect, the *EventServer* and the *AuthorServer*.

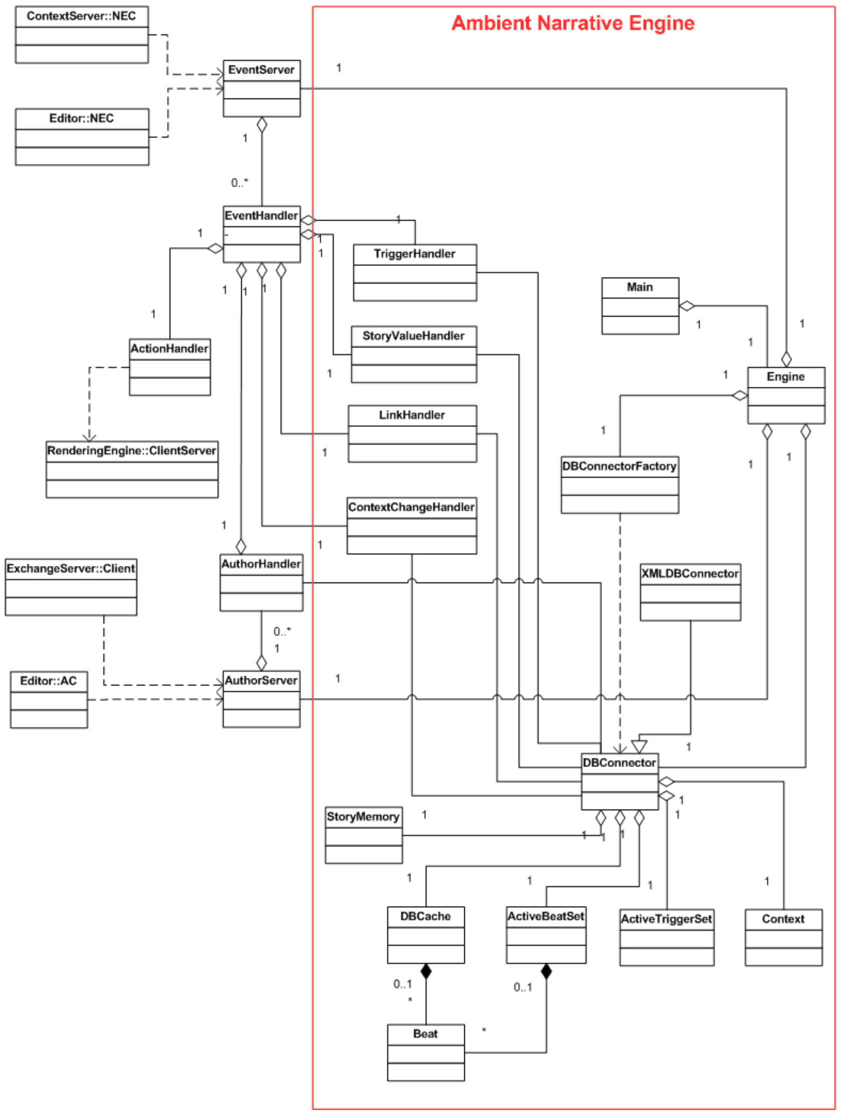


Figure 5.2: Static structure diagram Ambient Narrative Engine component

The *EventServer* accepts requests from clients interested in sending logical context events and/or receiving notifications of changes in the (active) beat set and creates a separate thread (*EventHandler*) for each client. The *AuthorServer* waits for incoming requests of clients who want to modify the beat database. For each author client a new thread (*AuthorHandler*) is created. In addition, the *Engine* class creates a *DBConnectorFactory* class that will instantiate a connection to a database for storing and retrieving context and beat information. To support different database implementations without changing code, each specific database connector must implement an abstract interface called *DBConnector*. In the current implementation only an eXist DBMS database connector is supported (*XMLDBConnector*).

The *DBConnector* controls access to the context and beat database and caches the ambient narrative state for classes that perform operations on this state. The data classes *Context*, *StoryMemory*, *DBCache* (beat set), *ActiveTriggerSet* (list of currently activated and set but unactivated triggers), *ActiveBeatSet* and *Beat* cover the main elements of the ambient narrative data model, i.e. Definitions 4.21, 4.22, 4.23, 4.24, 4.25 and 4.1. respectively. Data classes Pre (Definition 4.2), Link (Definition 4.20), Trigger (Definition 4.19) and StoryValue (Definition 4.12) are not shown in Figure 5.2. The other definitions in Section 4.1 are not implemented as special classes but as lists of key-value pairs where the key represents the attribute name and the value the attribute value.

The *EventHandler* class is responsible for beat sequencing. The *AuthorHandler* for run-time modification of the ambient narrative. The *ContextChangeHandler* class takes care of updating the context model. The *ActionHandler* class controls the communication with the rendering engine and sends action scripts. The *TriggerHandler*, *StoryValueHandler*, *LinkHandler* classes are used to process trigger, story value and link commands encountered during the beat sequencing or run-time authoring processes.

5.4.1 Procedure class mapping

Table 5.1 finally describes the mapping of the procedures onto the classes of Figure 5.2.

5.5 Concluding remarks

In this chapter we described the implementation of the ambient narrative engine and more specifically, the beat sequencing and run-time authoring algorithms as presented in the formal model of Chapter 4. Beat sequencing is treated as a retrieval problem where the physical context, session state and the trigger set represent the user query and the (active) beat set the data set. We discussed caching of the ambient narrative state for achieving real-time response times and heuristics for avoiding sequencing problems and race con-

Procedure	**Class(es)**
BEATSEQUENCE	*EventHandler*
UPDATESTORYMEMORY	*StoryValueHandler, DBConnector*
UPDATECONTEXT	*ContextChangeHandler, DBConnector*
UPDATEACTIVEBEATSET	*EventHandler, DBConnector*
CHECKPRECONDITIONS	*EventHandler, DBConnector*
RECOMPUTETRIGGERS	*LinkHandler, DBConnector*
UPDATETRIGGERLIST	*TriggerHandler, DBConnector*
SENDACTIONSCRIPTS, SendData	*ActionHandler*
ADDBEAT, REMOVEBEAT, EDITBEAT	*AuthorHandler, DBConnector*
LockState, UnLockState	*EventServer, AuthorServer*
NotifyListeners	*EventHandler*
RemoveDuplicates	*EventHandler*

All procedures mentioned in the text but not shown here are implemented by *DBConnector*.

Table 5.1: Mapping of procedures to classes

ditions during beat sequence cycles. We showed how run-time authoring is implemented by modifying hypertext graphs and how the setting and retracting of triggers can be used as an optimization strategy to control the size of the trigger list. In the next chapter we discuss the interfaces to this ambient narrative engine in the overall system architecture.

6

An End-user Programmable Shop Window System

The formal ambient narrative model described in Chapter 4 covers the subset of functional retail requirements that addresses the language features needed to describe beats and the processes to sequence and modify these modular ambient intelligence fragments at run-time. In this chapter we discuss the remaining requirements that were not discussed in Chapter 4 and specify the system architecture of an intelligent shop window system that can be programmed by end-users. We describe the interfaces to the *Ambient Narrative Engine* component discussed in Chapter 5 and show how this component is embedded in the overall system architecture.

This chapter is organized as follows: Section 6.1 presents an end-user software engineering approach for controlling the entire lifecycle of ambient narratives and provides an overview of the main components and interfaces between these components in the intelligent shop window system. The individual components in this system architecture and their interface to the end-user are further discussed in Sections 6.2, 6.3, 6.4 and 6.5.

6.1 An end-user software engineering approach

Experience design is a multi-disciplinary effort and therefore the needs of several stakeholders need to be taken into consideration in the design of an end-user programmable shop window system. Unless otherwise mentioned we refer to user roles in the following sections because one stakeholder can be e.g. both retailer and operator. Section 3.2.2 showed that designers had a preference for the 3D simulation environment for creating and testing smart retail environments whereas retailers saw the in-situ authoring environment using a PDA as more appropriate. As several participants commented they would want to use both end-user programming environments in combination, the intelligent shop window system system architecture should address both Requirements SE.1 (3D simulation) and SE.4 (in-situ using PDA).

Besides the need for an end-user programming environment that is easy to use and suitable for different types of end-users, efficiency is another important issue. An intelligent shop window is seen by a retailer as an investment to attract more people to his store. The benefit of this investment should outweigh the cost of the investment in the smart retail environment to create a positive return on investment. The total cost of the investment does not only depend on the costs of purchasing and installing the necessary hardware and software and programming the required behavior but also the operational costs of maintaining and updating the intelligent behavior of the shop window system over time. This implies that the entire life cycle of creating, testing, deploying, maintaining and updating the intelligent behavior of a shop window system needs to be taken into account, not just the create phase.

In professional software engineering a similar approach is taken: Integrated Development Environments (IDE) support software engineers in the entire software life cycle – blending specification, design, implementation, component integration, debugging, testing and maintenance into a tightly integrated, interactive environment. More recently, people have been looking at ways to bring the benefits of rigorous software engineering methodologies to end-users [Burnett, Cook & Rothermel, 2004; Burnett, Engels, Myers & Rothermel, 2007]. A similar integral approach has also been suggested by [Hardman, Obrenovic, Nack, Kerhervé & Piersol, 2008] for tools used in the media production chain. This chapter adopts a similar end-user software engineering approach to allow end-users to control the entire content life cycle of intelligent shop window ambient narratives in an easy and efficient way. Before the underlying system architecture and its components are discussed in detail, the ambient narrative content life cycle is seen from an end-user point of view.

6.1.1 Intelligent shop window life cycle scenario

To illustrate the ambient narrative life cycle we consider the intelligent shop window scenario. The designer will start off with an intelligent shop window installation that does not yet exist in the real shop. The designer therefore first needs to create a 3D model of the shop and the shop window. If the designer has finished constructing this 3D model of the shop using 3D modelling software in his office, he can import this model into the 3D simulation environment. He is then ready to add virtual sensors and actuators in the 3D environment that correspond with the real sensors and actuators used by the intelligent shop window system in the real store. If we assume the designer has taken these steps, he is ready to create the intelligent shop window ambient narrative using a visual editor that is connected to the 3D simulation environment as shown in Figure 6.1 (top left). The designer can test the behavior of the virtual intelligent shop window and simulate how the real intelligent shop window will appear and behave. The 3D simulation can be seen as a visual debugging tool as it enables the designer to correct errors in the intelligent behavior. If the designer is satisfied with the result he can show the simulated intelligent shop window ambient narrative to the store owner for feedback and make adjustments if needed. If the store owner approves, the designer can save the ambient narrative and upload it to a Web site or make a local copy.

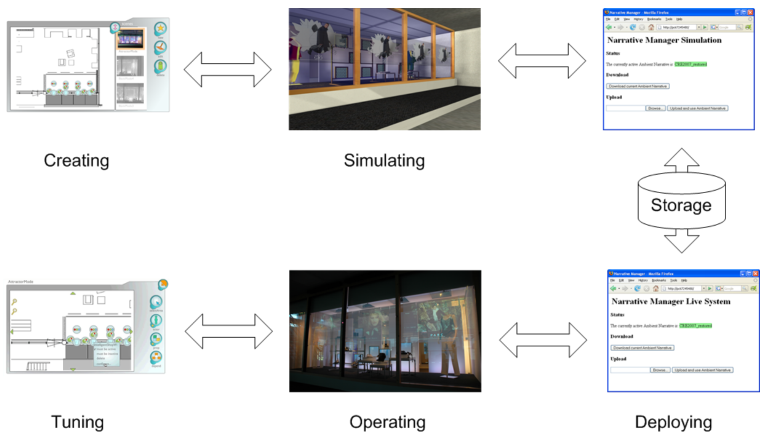


Figure 6.1: Intelligent shop window lifecycle scenario

If we assume the operator of the store has set-up the necessary hardware (physical sensors and actuators) and software for the intelligent shop window installation, the operator can download the intelligent shop window ambient narrative from the designer's Web site and deploy it onto the live system for operational use. Customers can now physically experience the behavior of the intelligent shop window as was discussed in Section 3.3.1. If the designer wants to maintain or update the behavior of the ambient narrative he can use a visual editor connected to the live system. The designer can immediately experience the effects of his actions in the real world, making it possible to tune behavior that was difficult to simulate or update the intelligent shop window ambient narrative to accommodate for changes in the collection or upcoming events, Figure 6.1 (bottom left).

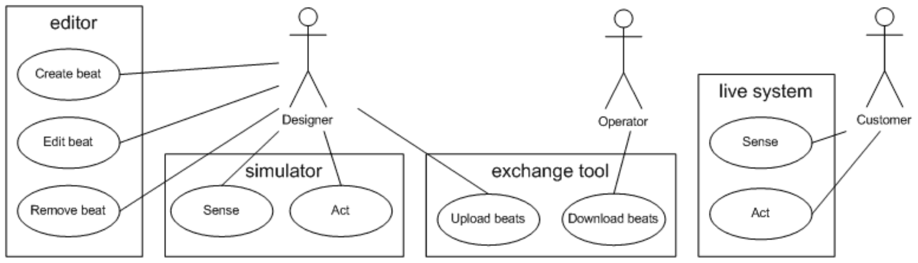


Figure 6.2: Use case diagram

Figure 6.2 shows the UML use case diagram of the user scenario discussed above. Using a 3D simulation the designer can explore the virtual shop window and interact with the 3D environment. The designer can create, edit and remove beats using a visual editor. The designer and operator use an exchange tool to upload and download ambient narratives between the 3D simulation and the live system. The customer explores and interacts with the live intelligent shop window system.

6.1.2 System architecture

Figure 6.3 presents the system architecture. To ease portability of ambient narratives (Requirement SE.2) the 3D simulator (Section 6.3) and live system share a common run-time system component. This component could be shared by both systems, blending real and virtual sensors and actuators in one system, or instantiated separately as in the user scenario above in which case there need to be two instances of the common run-time system component, one connecting to the real sensors and actuators in the physical store, the other using virtual sensors and actuators in the 3D simulation. The run-time system component can be further decomposed into the *Context Server*, *Ambient Narrative Engine* and *Rendering Engine* component. The *Context Server* (Section 6.2.1) aggregates and filters raw sensor data from the real or virtual environment. Important context changes are communicated to the *Ambient Narrative Engine* (Chapter 5, Section 6.2.2) that determines which ambient narrative beats need to be activated or deactivated. The action scripts of these beats are forwarded to the *Rendering Engine* (Section 6.2.3) that schedules the device actions in these scripts for rendering. Note that not all sensor data needs to be fed through the run-time system component, sensors can also be directly connected to actuators if the sensed data is not needed for determining the next action in the real or virtual environment and only used locally by an actuator. The *Editor* component (Section 6.4) allows the designer to view and modify the beat database of the *Ambient Narrative Engine* and view the context information sensed by the *Context Server*. The *Exchange Server* (Section 6.5) component enables users to import or export the beat database of the *Ambient Narrative Engine*.

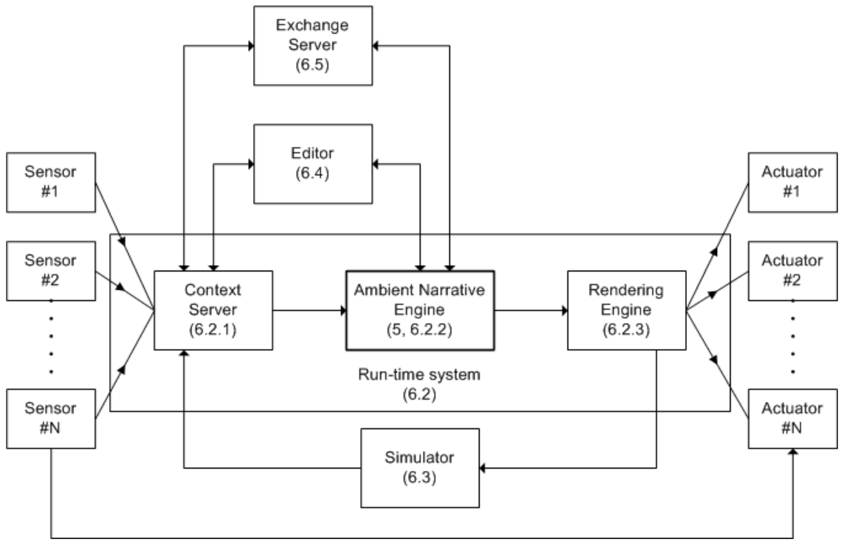


Figure 6.3: System architecture overview

The decomposition of the run-time system of a ubiquitous computing environment into sensing, reasoning and rendering components has also been proposed by others. The context framework for enabling the end user development of mobile context-aware applications implemented by [Korpipää, Malm, Salminen & Rantakokko, 2005] speaks of a context manager, rule script engine and activator component for sensing, reasoning and rendering, whereas [Clerckx, Vandervelpen, Luyten & Coninx, 2006] uses the terms context management system, dialog control and distribution manager to refer to these components. The ambient intelligence platform for physical play discussed by [Wakkary, Hatala, Lovell & Droumeva, 2005] uses a sensing system, reasoning engine and display engine. Service-oriented ubiquitous computing frameworks typically view the reasoning and rendering component as a single application layer that can build on top of services such as data storage, computing sharing, context management.

6.2 Run-time system

The run-time system provides real-world or simulated sensor input to the beat sequencing process of the Ambient Narrative Engine and processes its output, i.e. the resulting device action commands. The run-time system as a whole implements the following requirements RT.5 (*No noticeable difference in performance with a custom-built system*), RT.6 (*Scalable in terms of sensors and actuators*) and RT.7 (*Robust against sensor and actuator failures*).

To ensure Requirement RT.5 is satisfied the run-time system should respond in real-time as a tailor-made software application which would offer the same functionality. The potential bottleneck in the run-time environment is the central beat sequencing engine. Because this component reacts on sensor changes one way to reduce the system load is to filter out unimportant sensor events as early as possible. Two categories of unimportant sensor events can be distinguished: Events that will not cause a change in the active beat set and events that are only locally used by one or more actuators. Events that belong to the first category will not cause a change in the active beat set. To determine these events, the Context Server requires knowledge of the active trigger preconditions. Therefore there are two context models maintained in the run-time environment, a *physical model* of (changes in) the physical environment by the Context Server and a symbolic model of (changes in) the physical environment by the Ambient Narrative Engine. To deal with events which are never interpreted centrally, sensors can be coupled directly to the necessary actuator(s) as shown in Figure 6.3.

To cover a large shop, hundreds of sensors and actuators may be needed to obtain an accurate and precise model of the actors, props and performances that take place. The shop window area is typically much smaller so Requirement RT.6 can be relaxed. Furthermore, the user-generated ambient narratives found showed few dependencies between the post-its with beat descriptions. Therefore it is possible to divide a larger shop up into smaller shop segments, each having their own instantiation of the run-time system. Alternatively, a single Ambient Narrative Engine could receive input from multiple context server each addressing a segment of the shop, and/or on the actuator side, distribute output over multiple Rendering Engines. For the implementation of the end-user programmable shop window system, both the run-time system used by the simulation environment and the live environment only have one instantiation of each component in the run-time system.

To address Requirement RT.7 the different components in the run-time system need to reconnect automatically if a sensor or actuator (temporarily) disappears. This requirement is automatically satisfied for sensors and actuators if Requirement RT.8 (*Plug-n-play actuators*) and RT.9 (*Plug-n-play sensors*) are met and so will be discussed as part of Sections 6.2.3 and 6.2.1 respectively.

6.2.1 Context Server

The function of the Context Server is to filter and translate physical context changes into symbolic context changes for the Ambient Narrative Engine. To implement Requirements RT.1 (*React on implicit context changes*) and RT.2 (*React on explicit user feedback*) it is necessary to concentrate on a particular application domain and determine which types of sensors (and actuators) are needed. To support the intelligent shop window scenario domain of Section 3.3.1 the following sensors are used:

- ◇ Pressure sensitive floor tiles [Tekscan, 2008] in front of the shop window to identify and track a small number of people simultaneously, based on the pressure patterns of their feet.
- ◇ Ultra-wideband location system [Ubisense, 2008] to identify and track tagged objects, i.e. actuators and tangible objects of interest.
- ◇ Gaze tracking system [SmartEye, 2008] to detect where people are looking in the shop window.
- ◇ Optical touch screens [Ubiq'window, 2008] to detect where people are pointing on the display in the shop window.

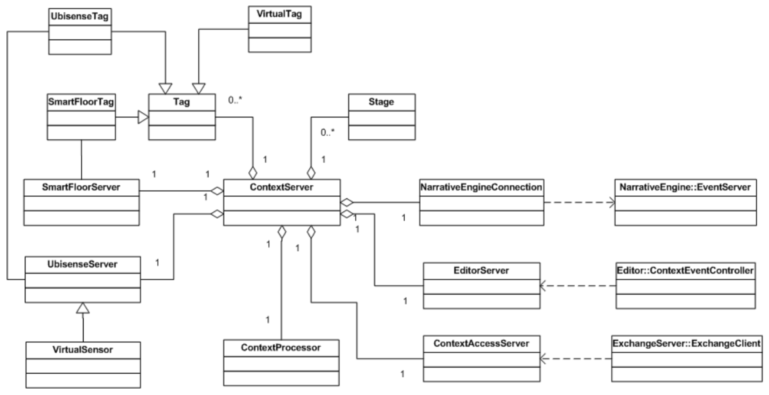


Figure 6.4: Static structure diagram Context Server component

User events from the gaze tracking system and optical touch screens sensors to control the interactive shop catalog on the shop window displays are handled directly by an intelligent shop window actuator (see Section 6.2.3) and therefore not further discussed here. The implementation of the sensors falls also outside the scope of this thesis.

Plug-n-play sensors(Requirement RT.9) are not supported by the current Context Server implementation. It is assumed the pressure sensitive floor and ulta-wideband location system are fixed in place during operational use as there is no functionality in the scenario indicating the need for a dynamically changing sensor infrastructure. Requirement RT.7 (*Robust against sensor and actuator failures*) is partially met by making sure the Context Server automatically tries to reconnect to sensors if the connection is lost. Requirement RT.8 (*Plug-n-play actuators*) is partially met by equipping each (portable) actuator with a tag that can be detected by the ultra-wideband location system. If this tag has an identifier that maps onto a prop that is used in the precondition section of a beat, the author can specify what should happen if a new actuator enters the stage.

Figure 6.4 shows the context server component in detail. The pressure sensitive floor system is accessed through the *SmartFloorServer*, the ultra-wide band location system through the *UbisenseServer*. The *ContextServer* creates a SmartFloorServer and UbisenseServer object to interface with these sensors to maintain a physical context model. This model implements the functionality needed to support the core ambient narrative language listed in Table 4.1; i.e. the absolute position and identity of actors and props on stages (presence performance only). Actors and props in the ambient narrative language model are both represented by *Tag* objects. Fixed objects are represented by *VirtualTag* objects. Each Tag object has a name, role, position and global attribute. The global flag can be set to indicate that this Tag is part of every defined Stage. Stages are described by *Stage* objects. Each Stage has a name and a set of absolute coordinates that define the area. A Tag can be present on zero or more Stages simultaneously and each stage object has a name that corresponds with the stage id attribute of a beat in the beat database managed by the Ambient Narrative Engine.

The ContextServer also instantiates a number of connections to other components. The connection with the Ambient Narrative Engine is maintained by a *NarrativeEngineConnection* object. The ContextServer also starts an *EditorServer* and a *ContextAccessServer* thread to provide an interface to the Editor and Exchange Server components respectively. Finally the ContextServer creates a *ContextProcessor* thread that filters and translates incoming physical context changes into symbolic context changes. For each Tag position change the ContextProcessor must notify editors subscribed to the EditorServer so they can update their visualization of the current context to the user. In addition, the ContextProcessor must check for each Stage if the Tag has entered or left this Stage. If a Tag has entered or left a Stage, the Ambient Narrative Engine must be notified of this change. Via the EditorServer, the ContextServer also receives commands from the Editor component to add a new Stage to the list of Stage objects or remove an existing Stage from this list. If this happens, the Ambient Narrative Engine component is also notified.

The separation between a physical context model and a symbolic context model can also be found back in other ambient intelligence systems. The MiddleWhere [Ranganathan, Al-Muhtadi & Chetan, 2004] distributed middleware infrastructure allows location-sensitive applications to be notified whether a certain location-based condition becomes true similar to our ContextServer and also maintains a spatial database to model the physical world that includes a physical layout of the environment as well as relevant physical objects. The ContextDistillery system presented in [Fujinami, Yamabe & Nakajima, 2004] makes a distinction between raw sensor data and more abstract, high-level context extractors (called distillers). Applications can use both raw sensor data and abstract high-level context information. In the shop window situation, the intelligent shop window applications are directly coupled to the gaze tracking and optical touch screen systems to receive raw sensor data but these applications are controlled in turn via the Ambient Narrative Engine that uses symbolic abstract context information described in the preconditions of a beat.

6.2.2 Ambient Narrative Engine

The Ambient Narrative Engine is responsible for calculating the next action in the physical or virtual environment depending on the available symbolic context information, session state and user feedback and plot material encoded by an ambient narrative, which can be modified at run-time. The implementation of this component was discussed in Chapter 5. In this section the interfaces to other components in the system architecture are discussed.

The interfaces to the Ambient Narrative Engine are depicted graphically in Figure 6.5. The *EventServer* waits for incoming requests of clients interested in sending symbolic context changes or receiving changes in the active beat set. If a client connects, the EventServer starts a new thread (*EventHandler*) to service this client. One of the clients that connects to this EventServer is the *NarrativeEngineConnection* of the Context Server. Through the NarrativeEngineConnection symbolic context changes are sent to the Ambient Narrative Engine. The Editor component connects using the *NarrativeEvent-Controller* to the EventServer of the Ambient Narrative Engine to subscribe

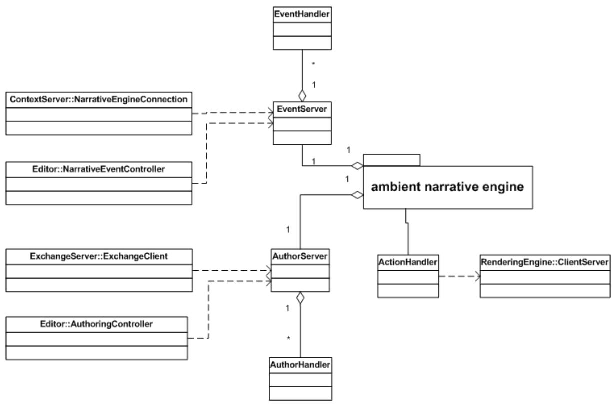

EventHandler
ContextServer::NarrativeEngineConnection
EventServer
Editor::NarrativeEventController
ambient narrative engine
ExchangeServer::ExchangeClient
AuthorServer
ActionHandler
RenderingEngine::ClientServer
Editor::AuthoringController
AuthorHandler

Figure 6.5: Static structure diagram Ambient Narrative Engine component (overview)

to changes in the active beat set.

The action scripts in the ‘init’ and ‘main’ sections of the beats that have been activated and the action scripts in the final section of the beats that have been deactivated as the result of the beat sequencing process in the Ambient Narrative Engine are forwarded to the *ActionHandler* that has a connection with the *ClientServer* of the Rendering Engine component.

Clients interested in modifying the beat set (i.e. add, remove or modify beats) connect using the *AuthorServer* to the Ambient Narrative Engine. For each client the AuthorServer starts a new thread (*AuthorHandler*) to service this client. The Editor component connects through the *AuthoringController* to the Ambient Narrative Engine. The Exchange Server component binds to an AuthorHandler object using its *ExchangeClient*.

Remarks

Session state changes are mostly caused by internal changes in the state of the ambient narrative as a result of story values that are being added or removed in the ‘init’ or ‘final’ sections of beats. Components may also send story memory changes to influence the session state and beat sequencing process by connecting to the EventServer of the Ambient Narrative Engine. In principle this would allow stateful actuators to set certain variables (and influence the beat sequencing process) based on the outcome of a device action command they received from the Rendering Engine component. Although this functionality is supported, it is not used for two reasons. First, it is not needed for the functionality of the intelligent shop window scenario. Second, it makes it harder to debug errors because the undesired behavior can now be caused by a story value change in one of the beats in the ambient narrative or in one of the sensors or actuators used by the system (and may not even be viewed inside!).

Explicit user feedback can similarly be modelled by a session state change or by a sensor in the Context Server that results in a symbolic context change forwarded to the Ambient Narrative Engine. For example, if a person presses a button in a user interface to register himself as being present on a particular stage that action corresponds to that person being detected by a location system on that stage. In other words the difference between explicit user feedback and implicit user feedback disappears at the symbolic level.

6.2.3 Rendering Engine

The Rendering Engine component takes care of the timing and synchronization of actions over multiple devices (Requirement RT.4). For the intelligent shop window scenario of Section 3.3.1 the following actuators are used:

- ◇ PC running a custom-made Flash application that controls what is displayed on the transparent holographic rear projection screen [DNP, 2008] hanging in the shop window. This application has three different shop window modes that can be set (blank, demo and interact). Blank mode turns the display black (transparent), demo mode sets a predefined product presentation slideshow and interact mode sets the predefined interactive shop catalogue. The gaze tracking system and optical touch screen connected directly to this application. For each shop window this set-up is replicated.
- ◇ DMX RGB wallwashers [ColorKinetics, 2008] to light up the back wall. The color and intensity of the light can be set.
- ◇ DMX RGB light tiles [ColorKinetics, 2008] for highlighting products in the shop window. The color and intensity of the light can be set.
- ◇ DMX RGB spots [ColorKinetics, 2008] to highlight the mannequins in the shop window. The color and intensity of the light can be set.

The gaze and touch input is only locally used to control the content on the shop window so it was decided to bypass the ambient narrative engine to lower the number of events received.

Figure 6.6 describes the static structure of the Rendering Engine component. The *RenderingEngine* has a *DeviceServer*, *ClientServer* and *Timer*.

The DeviceServer enables actuators to register themselves. Each actuator that wants to be recognized and used by the Rendering Engine must be subclass of *Device* and register to this DeviceServer. The interface to the PC application that controls the shop window is for example described by the *Shop-*

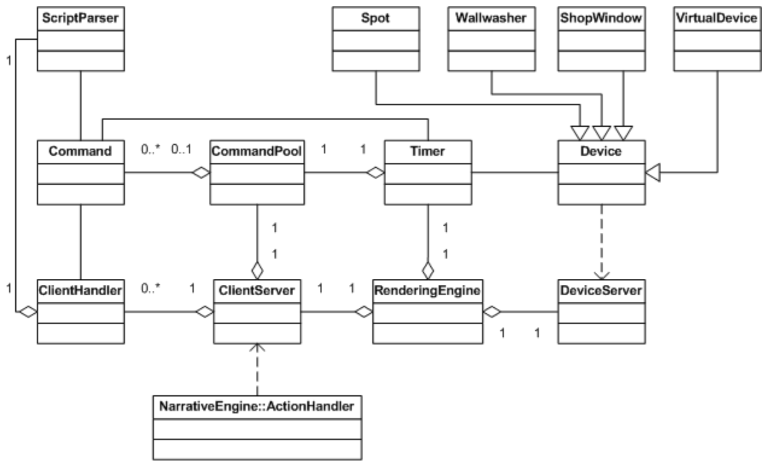


Figure 6.6: Static structure diagram Rendering Engine component

Window class that registers itself with the DeviceServer. After registration an actuator is ready to receive device action commands. This completes Requirement RT.8 (*Plug-n-play actuators*). Requirement RT.7 (*Robust against sensor and actuator failures*) is fully satisfied by making sure not only the sensors but also the actuators reconnect, in their case to the Rendering Engine if their connection is temporarily lost.

The ClientServer accepts connection requests from the *ActionHandler* in the Ambient Narrative Engine component. For each ActionHandler that connects the ClientServer creates a *ClientHandler* object to service this client. The ClientHandler uses the *ScriptParser* to parse received action scripts. This creates a list of device action command objects that either need to be added or removed for rendering. Each Command object has a script name, device name, action value, action z-order, timestamp and start time attribute. Command objects are added to or removed from the *CommandPool*. The CommandPool schedules, orders the device action commands based on the timestamps and start times.

The Timer controls the timing of device actions. It periodically checks the CommandPool for Command objects that need to be activated because their scheduled start time matches the current time and then sends the value and z-order of these device action commands to the appropriate actuator for rendering.

To conclude the discussion on the run-time system component we present the sequence diagram of handling a symbolic context event by the different components in the run-time system (Figure 6.7).

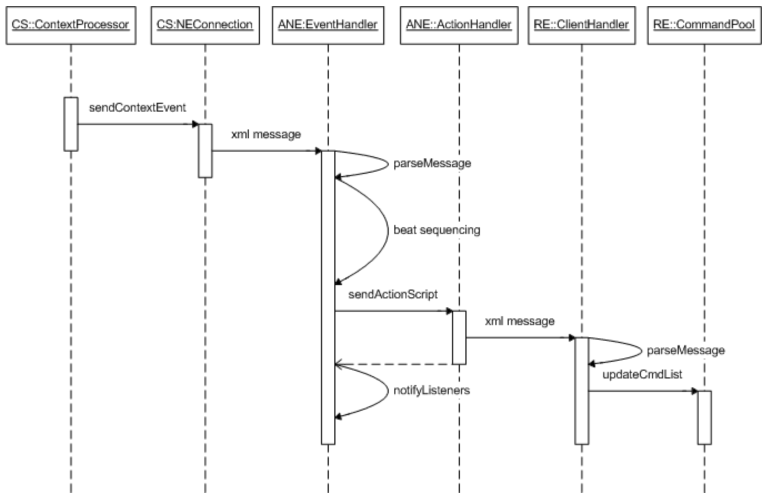


Figure 6.7: Sequence diagram handling symbolic context change

If the ContextProcessor of the Context Server component has determined a symbolic context change that needs to be communicated to the Ambient Narrative Engine, the ContextProcessor invokes the *sendContextEvent* method of the NarrativeEngineConnection. This method sends an XML message containing the context change to the EventHandler of the Ambient Narrative Engine that serves this object. The EventHandler parses this message and calculates the changes on the active beat set as part of the beat sequencing process. The EventHandler calls the *sendActionScript* method of the ActionHandler to send the action scripts to the Rendering Engine. At the same time the EventHandler notifies all listeners interested in receiving active beat list changes (not shown further). The ActionHandler sends the action script as XML message to the ClientHandler of the Rendering Engine that serves the ActionHandler. The ClientHandler parses the message for device action commands that need to be scheduled and invokes the *updateCmdList* method on the CommandPool object.

6.3 Simulator

The behavior of the intelligent shop window installation can be simulated with the 3D Simulator component. This component fulfills Requirement SE.1 (*Simulation of the behavior of ambient narratives*) and also addresses Requirement SE.5 (*Debugging the behavior of ambient narratives*) because the 3D Simulator enables the designer to correct errors quickly without the need to test on the shop window system in the shop. Previous work by [Barton & Vijayaraghavan, 2003; Shirehjini & Klar, 2005] also reports simulators for ubiquitous computing. Ubiwise [Barton & Vijayaraghavan, 2003] provides both a first person view of the physical environment using the Quake III Arena game engine and a close-up view of devices a user may manipulate. Besides their use for rapid prototyping of context-aware environments, both Ubiwise and 3DSim [Shirehjini & Klar, 2005] offer a context visualization mode where instead of virtual sensors, real sensors can be used to test the performance of the sensor hardware. Our Simulator is implemented using the open-source CrystalSpace [CrystalSpace, 2008] Software Development Kit. This SDK can be used to create real time 3D graphics and is especially suited for building games. The focus of this section is on the interface to the run-time system and the application that uses the CrystalSpace SDK. Figure 6.8 describes the static structure diagram of the 3D Simulator.

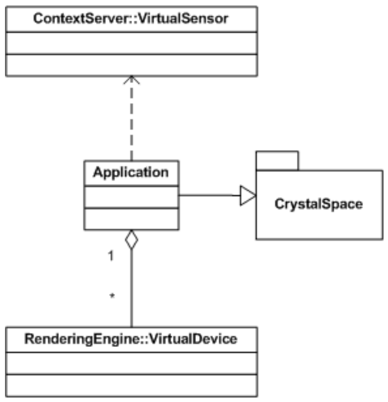


Figure 6.8: Static structure diagram Simulator component

The CrystalSpace *Application* uses the CS SDK to create a 3D simulation of ShopLab at the High Tech Campus in Eindhoven where the live intelligent shop window system is situated. The designer can walk through this 3D environment by moving his virtual character with the cursor keys and choose different perspectives to see how the simulated environment appears from different angles as shown by Figure 6.9.

The 3D model of ShopLab and the virtual devices and objects in these screenshots have been created using 3D Studio Max [Autodesk, 2008] and imported into custom CS model and scene files. The CS Application loads these files, creates the 3D world and the implements basic functionality to navigate and interact with this 3D world. The game logic that defines how the 3D environment reacts to symbolic events is described by the beats in the ambient narrative. The compiled source code of the physical layer in the CS

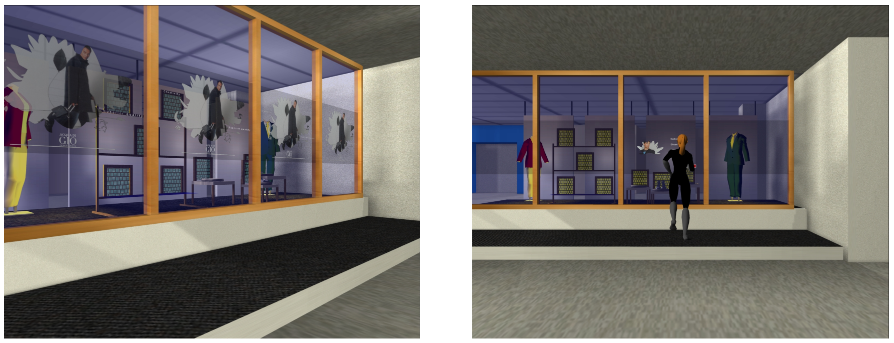


Figure 6.9: Screenshots of the 3D simulation of the shop window system

SDK implements the other necessary functionality such as e.g. rendering, object collision and motion detection support. A similar separation between game logic and game engine specific functionality is discussed by [BinSubaih, Maddock & Romano, 2005]. In the remainder of this section we concentrate on the interface with the ambient narrative run-time system component and game logic. For more information on the CS physics libraries we refer to the CS documentation and source code.

To simulate the intelligent shop window scenario, the virtual environment needs to implement virtual versions of the sensors and actuators that are used in the real environment. When the CS Application is started, it loads in 3D models of the actuators and instantiates *VirtualDevice* objects that can then receive device action commands from the Rendering Engine. If a VirtualDevice receives a device action command, it uses the CS libraries to render the effect of this action in the virtual world, e.g. change the image on the shop window display as shown in Figure 6.9. Changes in the position of the virtual character (actor) and actuators (props) are forwarded by the CS Application to the *VirtualSensor* of the Context Server that filters and translates these simulated physical context changes into symbolic context changes for the Ambient Narrative Engine. The frame rate of the simulation determines how many times per second changes in the virtual world are communicated.

The combination of virtual sensors and virtual actuators in the 3D Simulator corresponds to the default simulation mode or rapid prototyping mode. By interacting in the virtual world, virtual sensors detect virtual events that trigger beats that affect the virtual actuators. In case the virtual sensors are replaced by real sensors connected to the live shop window system, an operator can test the performance of the sensor hardware infrastructure to localize malfunctioning hardware/software or view in real-time the status of a live shop window system installed somewhere else. The advantage over e.g. security cameras is that the operator can view the status of the remote live shop window installation from different angles without affecting the live system. In the situation of virtual sensors and live actuators the Simulator turns into a tool for the operator to test actuator hardware or remotely control a live shop window installation like a puppet master in a puppet theatre play. This has the advantage that the operator can simulate sensor events such as the emotions of people that would be hard to detect by a sensor. When both the sensors and actuators are replaced by real versions the Simulator becomes a visualization of a live shop window system that cannot be controlled by the operator.

6.4 Editor

Using the Editor component users can modify the behavior of the shop window ambient narrative. The Editor needs to implement the user interface requirements UI.1 to UI.8. First the design decisions of the user interface are discussed and the layout and screens of the user interface then the interface of this component to the other components in the system architecture is discussed in detail.

6.4.1 User interface

Before users can add or delete beats in an ambient narrative they have to know what the current state of the ambient narrative is. In order to fulfill requirement UI.1 (*Summary of active and inactive ambient intelligence fragments*) the user should see be able to quickly obtain an overview of currently active and inactive beats and be able to identify them quickly. The difference between active and inactive fragments must be clear to the user. To improve identification of beats, each beat can be given a name and a photo can be taken. This photo is used a thumbnail image in the beat list.

The state of the ambient narrative is determined by the physical context. To meet requirement UI.2 (*Summary of current context situation*) the physical context model maintained by the Context Server must be visualized to the user. This helps the user to understand what the system is detecting and quickly figure out if a sensor or actuator is malfunctioning. Active maps that dynamically update the location of people, places and things have also been used by e.g. [Li, Hong & Landay, 2004; Barrenho, Romão, Martins & Correia, 2006] for mobile location-aware services.

Figure 6.10 shows the overview screen. The list of active and inactive beats is displayed on the right, the overview of the context situation, including a top-view map of the shop on the left. The user interface shows a number of

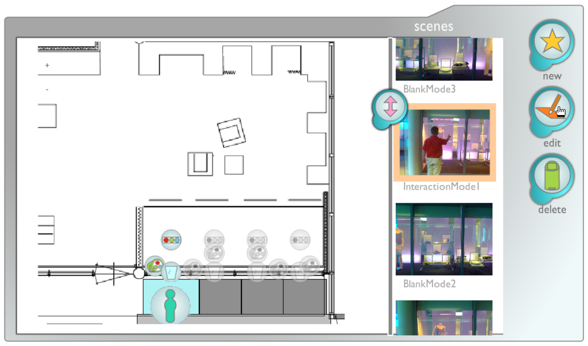


Figure 6.10: Overview screen

beats that are currently active including a beat named *InteractionMode1* that has been selected. The stage, actors and props in the precondition of the selected beat are superimposed over the context situation panel. The advantage of this connection between the selected beat and the context situation is that the user can quickly see why a beat might be active or inactive.

In the overview screen the user can choose to add a new beat, edit an existing beat or delete a beat from the list by pressing one of the buttons on the right. If the user presses the *New* button or *Edit* button the beat screen appears to edit the (empty) beat. If the user presses the *Delete* button the user is asked to confirm this action and the beat is removed from the beat set.

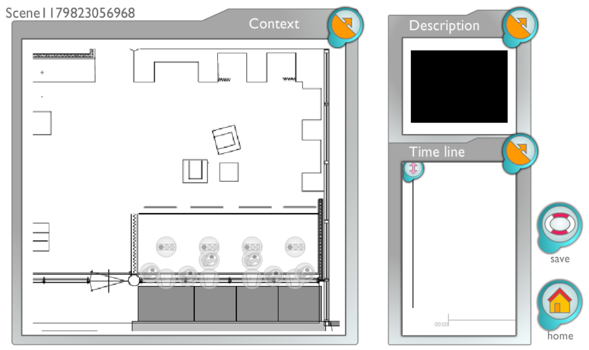


Figure 6.11: Beat screen

The beat screen provides a more detailed overview of the beat. The user can see a visualization of the beat's preconditions, the associated device actions and its name and thumbnail image. Figure 6.11 shows the beat overview screen if the user has pressed the new button. An empty beat with a new name is created that has no preconditions, no device actions and no thumbnail image defined. If the user selects an existing beat to modify the information in the beat is used. The user can modify the beat preconditions by left clicking on the icon next to *Context*. This brings up the beat precondition screen. The user can view the associated device actions by left clicking on the icon next to *Timeline*. This opens the beat action timeline screen. The user can change the name and thumbnail image of the beat through the beat description screen that is accessed by left clicking on the icon next to *Description*.

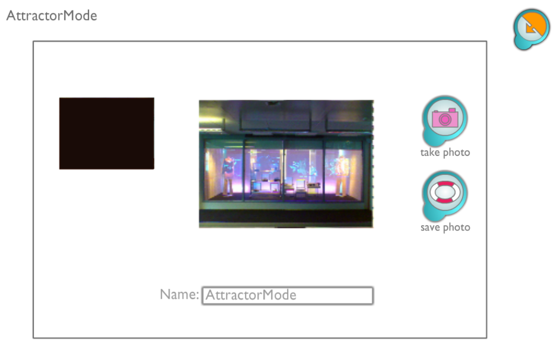


Figure 6.12: Beat description screen

If the user is satisfied with the result, the beat can be saved and added to the beat database in the Ambient Narrative Engine by left clicking on the *Save* icon. Should the user wish to undo actions he can always go back to the beat overview screen by pressing the *Home* button.

To complete requirement UI.3 (*Identification of ambient intelligence fragments*) the user must be able to type in a name for a beat and take a photo of the effects of this beat on its environment. This is done in the beat description screen, Figure 6.12. The user can change the name *AttractorMode* and take a photo by pressing the *take photo* button that is used as thumbnail image in the overview and beat screens.

The beat preconditions screen shown in Figure 6.13 implements Requirement UI.4 (*Overview of ambient intelligence fragment situation*) and Requirement UI.6 (*Control over the activation of ambient intelligence fragments*). Using this screen users are able to view and modify preconditions and thereby define how a beat is triggered. Visual rule-based approaches for authoring mobile location-aware applications and augmented reality have been described by [Güven & Feiner, 2003; Sohn & Dey, 2003; Li, Hong & Landay, 2004]. In the 3D authoring environment discussed by [Güven & Feiner, 2003] users can create locations and associate snippets (media objects with start and end times, position and other data). iCAP [Sohn & Dey, 2003] allows end-users to

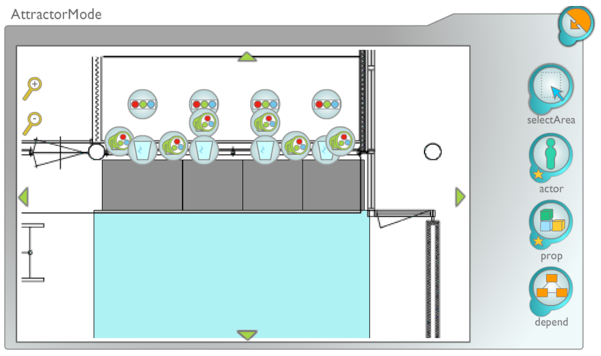
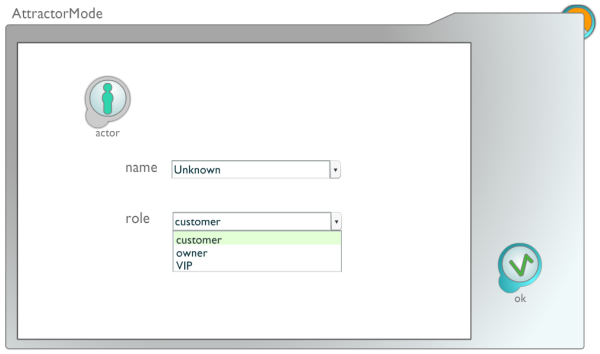


Figure 6.13: Beat preconditions screen

drag user-defined inputs, outputs, locations and people on a situation area to construct a rule. An alternative approach is to combine the input and output of components together as pieces of a puzzle or mashup. This jigsaw metaphor is used by [Rodden, Crabtree & Hemmings, 2004; Ballagas, Memon, Reiners & Borchers, 2007; Yahoo, 2008]. Viewing human computer interaction as theater has been proposed by [Laurel, 1986]. The user is seen as the audience, the agents in the user interface as actors and the desktop as being the stage. In [Finzer & Gould, 1993] programming by demonstration is looked at as rehearsing a performance. In [Repenning & Sumner, 1994] the authors argues for a participatory theatre metaphor where the audience, the user is not a passive observer but an active participant in the play and can choose his level of engagement.

The beat preconditions screen combines a visual rule-based language with a participatory theatre metaphor: The user can create or modify a stage by pressing the *selectArea* icon on the right. This allows the user to draw a rectangle on the map that represents the stage. To add an actor to this stage the user must press the *actor* button on the right. This opens the actor screen. To add a prop to this stage the user can left click on one of the device icons on the map or open up the prop list screen by pressing the *prop* button to select a prop from the list to configure. To set story values that must be tested, the user must press the *depend* button to bring up the dependencies screen. If the user is satisfied with the result he can press the navigation button in the top right corner to return to the beat screen (and view or save the beat).

The actor screen is used to set restrictions on the name and role of an actor. The actor screen is opened by pressing the actor button or left clicking on an existing actor icon on the map in the beat preconditions screen. Figure 6.14 displays the actor screen.



name Unknown

role customer

customer
owner
VIP

Figure 6.14: Actor screen

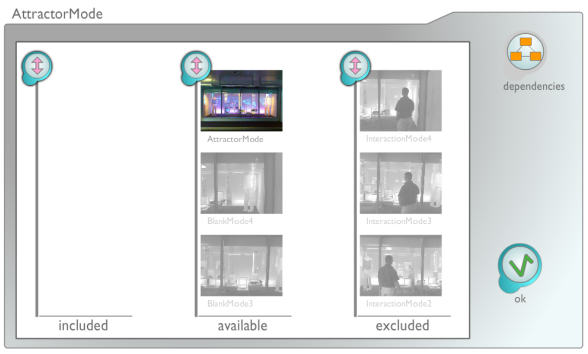
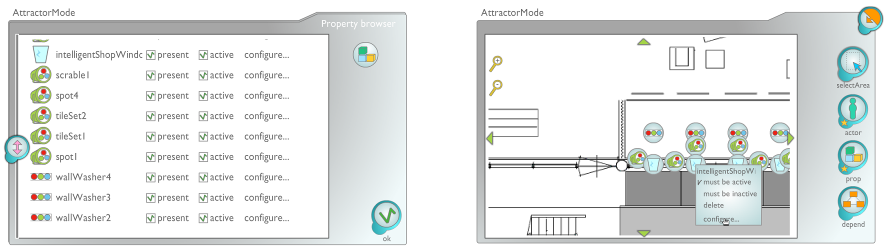


Figure 6.15: Dependencies screen

The dependencies screen (Figure 6.15) can be used to define which beats must be active and which beats should not be active for this beat to trigger. The Topiary storyboard workspace [Li, Hong & Landay, 2004] offers similar functionality to end-users to create dependencies between defined scenes. In the intelligent shop window scenario of Section 3.3.1 the product presentation slideshow is shown unless a person is standing directly in front of one of the shop windows, in which case the interactive shop catalogue should be presented and the lighting adapted. To model this behavior the beat that describes the product presentation slideshow (*AttractorMode*) must test on the non-existence of beats that describe what should happen when a customer is standing in front of one of the shop window displays (*InteractionMode4* etc.). Otherwise both beats would be active simultaneously because both preconditions would be met. This is done by moving beats from the *available* into the *excluded* list in Figure 6.15. Beats that must already be active before this beat can be triggered can be moved to the *included* list.



intelligentShopWindc	✓ present	✓ active	configure...
scrable1	✓ present	✓ active	configure...
spot4	✓ present	✓ active	configure...
tileSet2	✓ present	✓ active	configure...
tileSet1	✓ present	✓ active	configure...
spot1	✓ present	✓ active	configure...
wallWasher4	✓ present	✓ active	configure...
wallWasher3	✓ present	✓ active	configure...
wallWasher2	✓ present	✓ active	configure...

Figure 6.16: Two ways of selecting and configuring props

The prop list screen is shown in Figure 6.16 (left). The user can choose a prop in the list to configure a device action and set whether this prop must be present, i.e. included on the stage and if the prop must be active or not. This latter option allows the user to test on the non-existence of a prop and trigger a beat when a device is switched off or taken away. Although the prop list can be used to configure device actions, it is often easier and faster to left click directly on a prop on the map in the beat precondition screen to configure it as shown in Figure 6.16 (right).

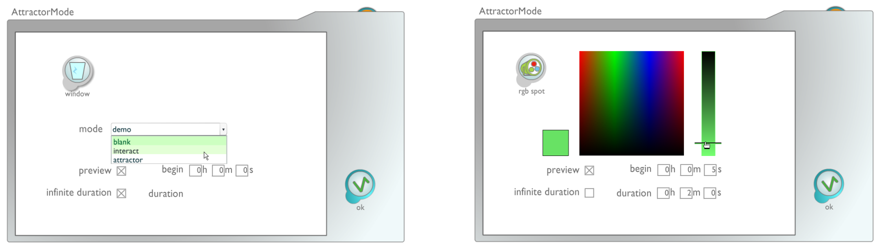


Figure 6.17: Prop editors

If a user chooses to configure a prop, the corresponding prop editor is started. This enables the user to set a device action and its start time and duration. This fulfills Requirement UI.8 (*Control over the action content in ambient intelligence fragments*). Because of the variety between devices and applications, each prop implements its own device action editor that is loaded by the main editor when the prop is detected by the system. Figure 6.17 shows two prop editors that are used to compose the intelligent shop window scenario, one to set the application mode on the shop window display (left), the other to set a RGB spot light (right).

The beat timeline screen shown in Figure 6.18 fulfills Requirement UI.5 (*Overview of ambient intelligence fragment action*) and UI.7 (*Control over the timing and synchronization of actions in fragments*). In the action timeline the user can see the start and stop times of device action commands in one single screen and modify these timelines by left clicking on the prop icon. This brings up the associated prop editor screen to modify the device action. Only one device action can be set for each device. To preview the device actions associated to the beat before saving, the user can press the play button at the right. This functionality is useful in the in-situ authoring environment where a lighting designer can try out different light settings in the shop without having to first stand in front of the shop window for example to trigger the beat.

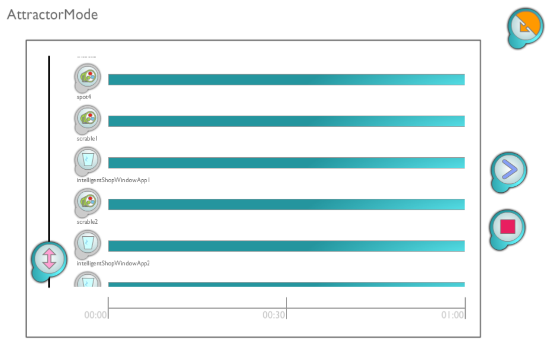


Figure 6.18: Beat timeline screen

Figure 6.19 graphically depicts the navigation structure between the screens of the editor user interface.

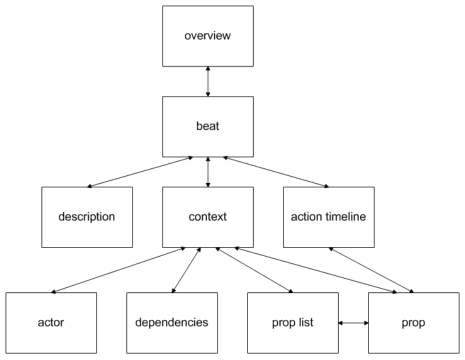


Figure 6.19: User interface flow diagram

6.4.2 System interface

The Editor component and its interface to the Context Server and Ambient Narrative Engine component is shown in Figure 6.20. The Editor component implements a standard model-view-controller software architecture pattern.

The controller part is implemented by the *Editor* and its helper classes in the upper part of Figure 6.20. The graphical user interface, view is realized by the *Main* class and related classes (lower part Figure 6.20). The beats of the ambient narrative and physical context model of the Context Server form the model part, they are visualized and manipulated by the controller through the user interface.

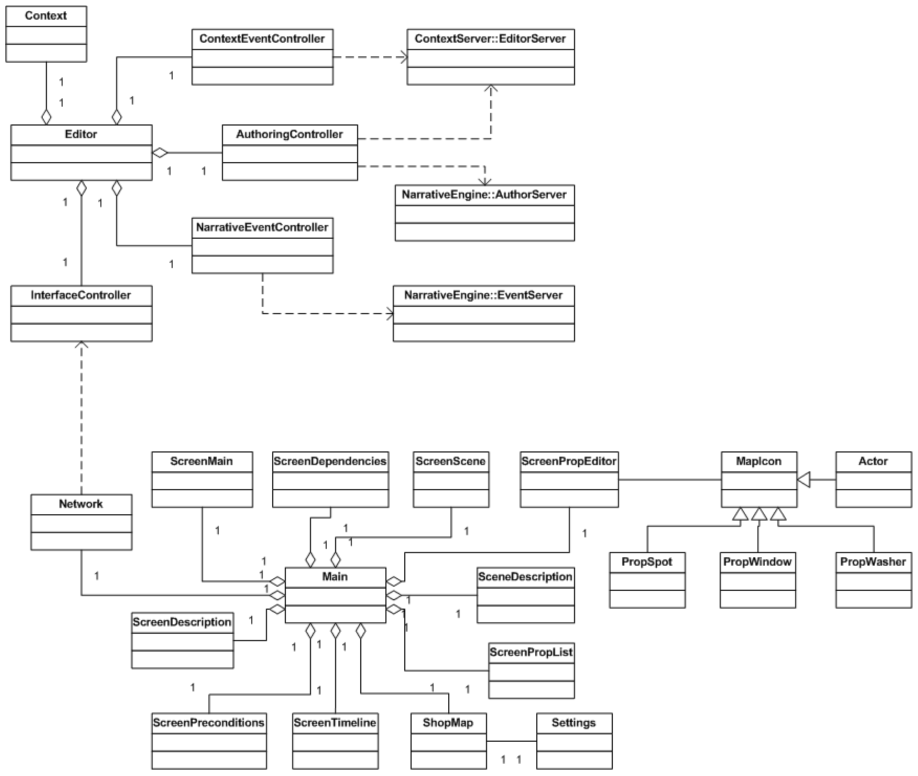


Figure 6.20: Static structure diagram Editor component

The Editor class starts the authoring tool application. It creates three separate threads to listen for changes in the physical context model (*ContextEvent-Controller*), changes in the beat set (*NarrativeEventController*) and changes in the user interface (*InterfaceController*). The ContextEventController connects to the *EditorServer* in the Context Server component to receive physical context change updates. The NarrativeEventController interfaces to the *EventServer* of the Ambient Narrative Engine to be notified of changes in the active beat set and other beats in the ambient narrative. The Editor maintains a local copy of both the physical context model (in a separate *Context* object) and this ambient narrative state (inside an NarrativeEventController object). The InterfaceController maintains the state of the user interface and sends changes to and receives instructions from this user interface. The InterfaceController is in control of changing the beat list and map in the overview screen (Figure 6.10) and instructing the *AuthoringController* to save a beat that has been created or modified in the user interface, remove a beat, save a photo and preview an action script. The AuthoringController connects to the *AuthorServer* of the Ambient Narrative Engine and the EditorServer of the Context Server.

The *Main* class of the user interface controls the activation of screens and creates a *Network* object to send and receive commands from the InterfaceController. The overview screen of Figure 6.10 is implemented by *ScreenMain*. Changes in either the physical context or beat set are directly forwarded while the user interface is in this overview screen state. If ScreenMain is activated, the context model and beat list are requested through the Network interface. The InterfaceController then sends the new list and context model. If the user presses the delete button, the selected beat identifier is sent to the InterfaceController for removal. The *ShopMap* and *Settings* are used to draw the map of the shop and the stages, props and actors in it. The beat screen is described by *ScreenScene*. The *SceneDescription* is used to parse and serialize a beat document. If the user presses the save button the newly created or modified beat object is materialized into an XML document that is forwarded to the InterfaceController to be added or replaced in the beat set. The *ScreenDescription* implements Figure 6.12. If the user presses the save photo button in this screen, the InterfaceController is instructed to save the sequence of photo pixels into a thumbnail image. The AuthoringController fowards this data stream to the Ambient Narrative Engine. The beat timeline screen is described by *ScreenTimeline*. If the user presses the play button, the action script object is materialized into an XML document that is fowarded to the InterfaceController for preview. The stop button is used to retract the previous script. Figure 6.12 is implemented by the *ScreenPreconditions* screen. The actor (Figure 6.14 and dependencies screen (Figure 6.15 are implemented by the *Actor* and *ScreenDependencies* classes respectively. The prop list of Figure 6.16 (left) is realized by the *ScreenPropList* class. The prop editors shown in Figure 6.17 are described by *PropSpot* and *PropWindow* and are subclasses of *MapIcon*. The *ScreenPropEditor* creates the prop screen and uses the MapIcon.

To fulfill requirement UI.9 (*Plug and write actuators*) the prop editor class URL of an actuator must be added as an attribute to the device Tag information that is kept by the Context Server. If a device is detected by the location positioning system or manually added, the prop editor URL will then be sent

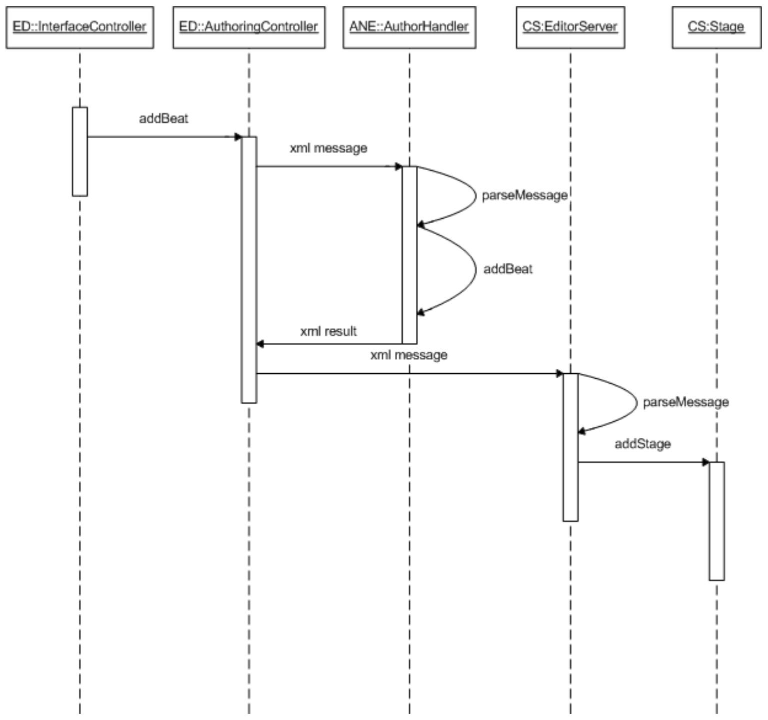


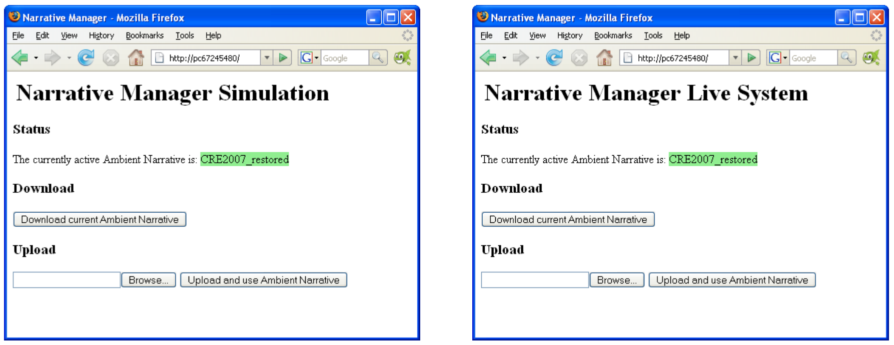
Figure 6.21: Sequence diagram beat addition

along with the other Tag data to the ContextEventController and the InterfaceController. If the user then selects the prop on the map or in the prop list, the corresponding prop editor class can be loaded. Requirement U.10 (*Plug and write sensors*) is not satisfied by the shop window system because the necessary plug and play functionality has not been implemented. The editor user interface has been limited to a fixed physical sensor infrastructure. In a dynamically changing sensor infrastructure situation the user interface must adapt continuously in order to provide feedback to the user that certain context situations cannot be detected by the present set of sensors.

To conclude this section we present a detailed description of how a beat is added to the shop window system. Figure 6.21 shows the UML sequence diagram of adding a beat. If the InterfaceController receives the instruction to add a new beat following a save event originating from the beat screen, the addBeat method on the AuthoringController object is invoked. The AuthoringController will serialize the beat object into an XML document that is handled by an AuthorHandler object in the Ambient Narrative Engine. The AuthorHandler parses the XML document and creates a beat object that is added to the beat set. The AuthorHandler acknowledges the result of this operation back to the AuthoringController. If the operation was successful, an XML message containing the name and coordinates of the stage is forwarded to the EditorServer of the Context Server. The EditorServer parses the message and adds the new stage to the list of Stage objects maintained by the ContextServer class. The new beat can now be triggered as a result of a context change as was shown in Figure 6.7.

The process of removing a beat from the system is similar to that of adding a beat, the main difference is that the stage is removed from the Context Server before the beat itself is removed from the Ambient Narrative Engine. This way the beat is automatically deactivated as part of the beat sequencing process and can then be safely removed.

6.5 Exchange Server


Narrative Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://pc67245480/
Download current Ambient Narrative
Browse...
Upload and use Ambient Narrative
Narrative Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://pc67245480/
Download current Ambient Narrative
Browse...
Upload and use Ambient Narrative

Narrative Manager Simulation

Status

The currently active Ambient Narrative is: CRE2007_restored

Download

Upload

Narrative Manager Live System

Status

The currently active Ambient Narrative is: CRE2007_restored

Download

Upload

Figure 6.22: Web interface to the Exchange Server

Figure 6.22 shows screenshots of the Web interface used to access the Exchange Server. Using this interface, users can upload or download intelligent shop window ambient narratives in the simulation (left) or live system (right) and save current and restore earlier made ambient narratives. The Exchange Server thereby meets Requirement SE.3 (*Versioning of ambient narratives*). To transfer an ambient narrative between the simulation and the live system, the user first has to download the current ambient narrative using the Exchange Server that is connected to the run-time system of the 3D Simulator. This brings up a file save dialog box asking him where the user wants to store the ambient narrative (e.g. on a shared network drive or portable USB stick). Next, the user has to switch to the Exchange Server that is connected to the live run-time system and press the browse button to select the ambient narrative he just saved. He is then ready to upload and use the ambient narrative in the live system.

The Exchange Server component and interfaces to the Context Server and Ambient Narrative Engine components are depicted in Figure 6.23. Through a *Web browser* the user accesses the *ExchangeClient* that is an application (PHP script) running on a *Web server*. This ExchangeClient provides a Web form (Figure 6.22) where users can choose to download or upload an earlier made ambient narrative. The ExchangeClient maintains a connection to the *ContextAccessServer* of the Context Server and the *AuthorServer* of the Ambient Narrative Engine component to import and export ambient narratives in the run-time system.

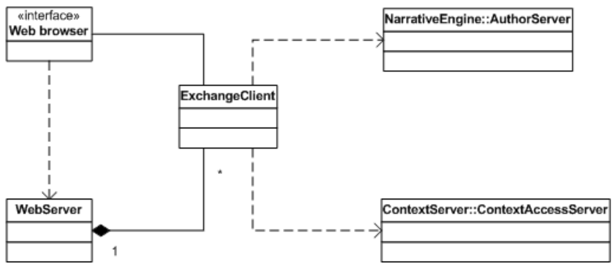

«interface»
Web browser
ExchangeClient
NarrativeEngine::AuthorServer
WebServer
ContextServer::ContextAccessServer
*
1

Figure 6.23: Static structure diagram Exchange Server component

If the geometry of the virtual world and real world match both the simulation and live environment can use the same physical context model used by the Context Server after scaling: In the shop window system implementation, the virtual 3D model is a scaled copy of the original building plan and therefore virtual coordinates can be translated into the real world physical context model that is used as the reference model. This situation corresponds with the case of an individual shop where the 3D simulation will be build to match the real world situation as closely as possible and this includes the geometry of the space. If the geometries are not similar two different physical context models need to be used. This occurs e.g. in a chain of shops where the geometry of the individual shops is likely to be different.

To download an ambient narrative, both the beat set of the Ambient Narrative Engine and the Stage list table of the Context Server need to be exported and saved to disk. The Tag list table of the Context Server does not need to be included as this part of the physical context model is automatically refreshed with new sensor data. The Stage list must be included because it contains physical context information that is associated with the beat set (Section 6.2.1). The ExchangeClient uses the AuthorServer to retrieve the beat files and the ContextAccessServer to fetch the stage list. The run-time system is temporarily stopped during this action to prevent inconsistencies between the stage list and beat set. The stage list files and beat files are packaged in a zip file.

Uploading an ambient narrative involves a number of steps. First the run-time system is paused to prevent changes in the beat set by other clients during the uploading process. Second, the Context Server is instructed to remove all tags from all stages and then removes all stages. This causes the Ambient Narrative Engine to deactivate all currently active beats, gracefully shutdown the running applications and effects and return to its default state. Third, the beats of the new ambient narrative are added. Fourth, the stage list of the Context Server is replaced by the new stage list. The Context Server forwards these stage changes to the Ambient Narrative Engine. Finally, the run-time system is restarted. The ContextProcessor of the Context Server component checks the Tag list now against the new stage list and forwards the symbolic context changes to the Ambient Narrative Engine.

Because an ambient narrative encodes how the actuators in the environment responds to symbolic events that take place in this environment it can be viewed as the game logic of a game engine as was discussed in Section 6.3. As described by [BinSubaih, Maddock & Romano, 2005] this game logic is portable and can be copied from one 3D Simulator to another using the Web interface to the ExchangeServer. The main purpose of the ExchangeServer is however to port this game logic to a *physical* game engine, the Ambient Narrative Engine component in the run-time system connected to real-world sensors and actuators.

6.6 Summary

In this chapter we presented the system architecture of an ambient narrative intelligent shop window prototype that enables retail designers to create, simulate, deploy and fine tune interactive multi-sensory shop window experiences. The architecture consists of a common run-time system component (of which the ambient narrative engine is a part) that provides an abstraction layer for sensors and actuators and that facilitates the portability of ambient narratives between the 3D simulation and the in-situ authoring environment. In addition it consists of an editor for creating and an exchange server component for versioning and deploying ambient narratives.

7

Evaluation

In the previous chapters the functionality provided by the end-user software engineering environment for writing intelligent shop window ambient narratives has been evaluated by building an actual prototype system. Although this implementation tells us something about the feasibility of our approach, more detailed analysis and further evaluation is needed to validate our proposed solution. In this chapter we revisit the prototype system and look at it from different angles.

Section 7.1 discusses the goals, set-up, method and results of a user study conducted to evaluate the usability of the system and find out to which degree end-users are able to design ambient intelligent shop windows using this system. Section 7.2 analyzes the technical feasibility, the strengths and weaknesses of the current system architecture by observations of use. The extensibility and generality of the prototype ambient narrative system and authoring environment are the topics in Section 7.3. Conclusions are presented in Section 7.4.

7.1 User evaluation

In evaluating the usability of a newly developed interactive application, the new application is typically compared with an application which already exists. This application provides the baseline that is used to measure the degree of improvement of the new system over the old one. In the absence of an earlier version, the ambient narrative prototype system cannot be benchmarked against such an earlier version. To test the usability of the prototype system it is therefore necessary to define evaluation criteria that capture the most important aspects on which we want to judge the usability of the system. The collected statistical data that is presented in this section should be interpreted qualitatively.

In [Resnick, Myers, Nakakoji & Shneiderman, 2005] a number of design principles are presented to guide the development of end-user programming environments. Among these guidelines for the design of creativity support tools is the criteria that an effective tool should have a *low threshold, high ceiling and wide walls*. Low threshold means that an effective tool should offer an interface that gives novice users the confidence that they can succeed in using the tool. High ceiling means that the tool is powerful enough to work on increasingly sophisticated projects. Wide walls means that a creativity tool should support and suggest a wide range of explorations. In other words, the user should be able to combine the features offered by the tool in many diverse ways. Because threshold is related to how intuitive a system is in use and ceiling and walls to the freedom and expressive power a system offers to its users we can derive two categories in which hypotheses need to be defined and tested to evaluate the usability of the ambient narrative prototype as a creativity support tool. A third category is efficiency: For end-users who need to use creativity tools as part of their everyday job it is important that the tool is not only intuitive and powerful but also efficient in use.

7.1.1 Hypothesis

The central hypothesis which was formulated as one of the research questions of this thesis and that needs to be verified in a user study is:

H0 Can end-users design ambient intelligent shop windows using the prototype system?

This hypothesis needs to be decomposed into a number of more specific hypotheses which fit into the categories of intuitiveness, expressiveness and efficiency that were defined earlier.

From the interviews and workshops discussed in Section 3.1) we learned that both retailers, designers and consultants can find themselves in the concept of a shop as interactive, improvisational theatre stage where both shop employees and customers are engaged in a performance that is partly scripted and that they could decompose user scenarios or storyboards into theatrical scenes or beats of an ambient narrative. The authoring tool is designed in such a way that the first screen users see when they start the application is an overview of the current context situation and the beats that are currently present and active in the ambient narrative. As a result we expect that the following hypothesis will hold:

H1 End-users can easily find an overview of the state of the intelligent shop window ambient narrative.

Buttons for creating new beats and modifying existing beats are directly next to the beat overview list. If a user presses one of these buttons he sees the beat overview screen where he can immediately see the actions and preconditions associated with this beat. During the workshop sessions we discovered that users placed the emphasis on specifying the desired action of a beat and paid less attention to the precondition part. We expect users having more difficulty in setting beat preconditions than beat actions. By following the shop as theatre metaphor in the precondition screen (define stage, actor, prop) and navigating users through this screen to add device actions to a beat we force the user to think about the conditions for activation of his beat. The notion of timelines used in the action screen is expected to be familiar to these users as it is commonly used in multimedia authoring tools. The hypothesis we want to verify is:

H2 End-users find it easy to create and modify the behavior of the intelligent shop window ambient narrative with the editor.

Expressive power is difficult to measure accurately with a user study as the freedom to express what the user has in mind to the ambient narrative prototype system depends on the sensors and actuators that are used and can be configured and on the implementation of the formal ambient narrative model. The intelligent shop window installation implements a small number of sensors and actuators and supports a minimal subset of the formal ambient narrative model of Chapter 3. Section 7.3 provides a more detailed technical analysis of the limitations that we placed upon ourselves but since users can only evaluate what they see we expect them to be critical about the possibilities they have to set beat preconditions and device actions:

H3 End-users consider the expressive power of the current system as restricted.

In terms of efficiency we can formulate the following hypothesis:

H4 End-users find the editor efficient in creating and modifying beats both in the live system and simulation.

Users can quickly navigate from the main overview screen to the precondition and action screens to set the preconditions and device actions of a beat. The editor used on the PDA is identical to the one used for the 3D simulation so we don't expect major differences between the two authoring environments in terms of usability.

7.1.2 Test set-up

The location for the user study was ShopLab which is part of the ExperienceLab, a usability and feasibility facility of Philips Research at the High Tech Campus in Eindhoven. ShopLab provides a realistic fashion shop environment that is used to test what end-users (shoppers, retailers) think about new retail concepts in an early stage in their natural setting. The intelligent shop window installation was integrated in the shop window area of ShopLab (Figure 7.2), the 3D simulation ran on a PC with dual monitors on a desk in a corner of the area in front of ShopLab (Figure 7.1). Participants were individually invited to ShopLab and after a short demonstration of both authoring environments that was the same for all subjects, each subject was instructed to perform a number of simple authoring tasks followed by a questionnaire that asked them to score both environments and allowed them to give feedback.

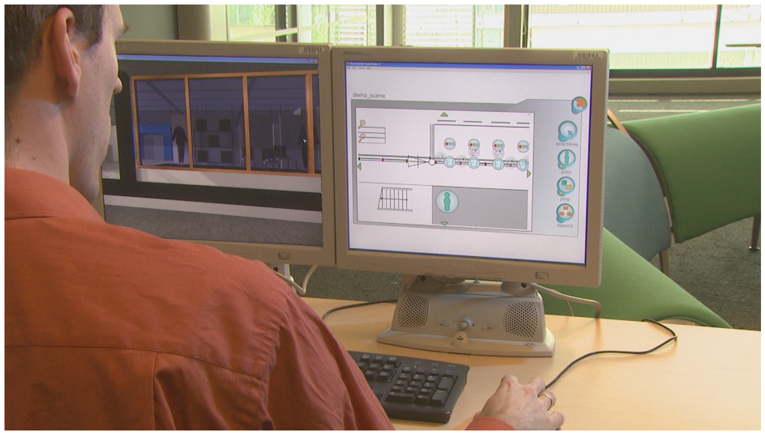


Figure 7.1: User study setup simulation

7.1.3 Participants

Most of the subjects (13 out of 18) that participated in the user evaluation had also participated in the interview and workshop sessions. Five participants were recruited afterwards. The participants had various backgrounds (retail management, interior design, industrial design, cultural anthropology), different levels of expertise (2-30 years), positions (employee, manager, execu-

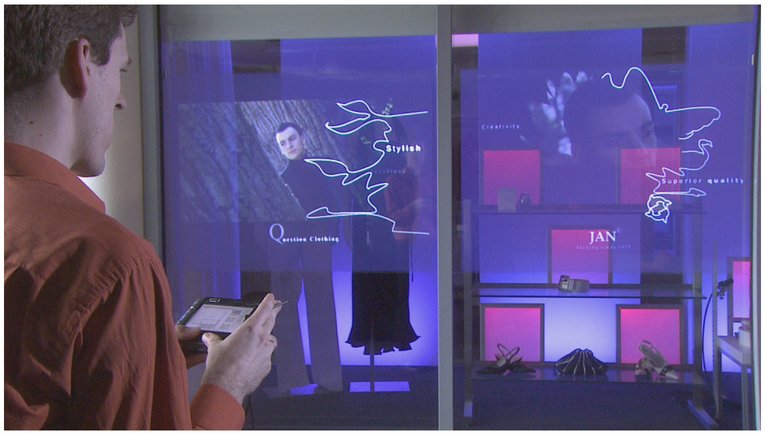


Figure 7.2: User study setup live system

tive, owner) and areas of expertise (retailer (27%), designer (35%), consultant (38%)). The majority of the subjects were Dutch (88%) and male (88%). 72% of all the participants had a retail background (education and/or experience). None of the subjects had a background in computer science or software engineering. Available time for the user study forced us to group subjects into one user category and prevented a more detailed analysis and comparison of the different types of stakeholders in the retail value chain.

User tasks

Each user had to perform two beat creation and two beat modification tasks in total. One creation and one modification task to evaluate the 3D simulation end-user programming strategy, the other set of two tasks to validate the programming on location strategy. The task descriptions are listed below. The word scene was used as a synonym for beat in the task description and questionnaire.

T1 **Scene modification in simulation:** Modify the 'demo' scene in the 3D simulation. Set the light tiles to red. (*Verander de 'demo' scene in de 3D simulatie. Zet de licht tegels (tile sets) allebei op een rode kleur*)

T2 **Scene creation in simulation:** Delete the 'demo' scene in the 3D simulation. Create the following scene in the simulation: When there is somebody (role customer) standing in front of the first shop window, set the window to 'interact' mode and the wall washer to blue. (*Verwijder de 'demo' scene uit de 3D simulatie. Maak de volgende scene in de 3D simulatie: Als er iemand (in rol customer) voor het eerste shop win-*

dow staat, moet het shop window in 'interact' mode en de wall washer een blauwe kleur krijgen.)

T3 Scene modification in-situ: Modify the interaction mode scene of the third window in the live system. Set the wall washer behind this window to green. (*Verander de interactie mode op het derde shop window in het live systeem. Geef de wall washer achter dit scherm een groene kleur.*)

T4 Scene creation in-situ: Delete all scenes in the live system. Create the following scene in the simulation: When there is somebody (role customer) standing in front of the right most shop window, set the window to 'interact' mode and the wall washer to red. (*Verwijder alle scenes uit het live systeem. Maak de volgende scene in het live systeem: Als er iemand (in rol customer) voor het eerste shop window staat, moet het shop window in 'interact' mode en de wall washer een rode kleur krijgen.*)

Procedure

To ensure all test subjects had the same prior knowledge a fixed procedure was followed: First, the intelligent shop window experience was demonstrated to participants by physically walking and interacting in front of the real shop window displays and seeing the shop windows react. The ambient narrative that described the behavior the participants experienced implemented the scenario described in Section 3.3 and was the same for all subjects.

After showing the user experience to people, the portable device with the editor to change the behavior of the live shop window was shown to participants. Subjects were told that the shop window behavior was composed of individual scenes that they could modify, delete and create. The features of the editor were then explained one by one (scene overview, current context situation overview, scene creation, scene modification, scene deletion, setting preconditions, setting device actions, setting scene name/taking a photo snapshot) by demonstrating a scene modification task that was the same for all participants and showing the effects in the live environment.

Third, participants were introduced to the 3D simulation set-up. Subjects were asked to imagine themselves being in the role of a designer sitting in the office to create the behavior of an intelligent shop window. The visualization features of the 3D simulation environment were demonstrated to participants starting with an empty ambient narrative (navigating around in the 3D environment; nothing happens). After this step the features of the editor were explained by demonstrating a fixed scene creation task and showing the result in the 3D simulation.

After this instruction, subjects were asked to switch places and perform the tasks that were described in the previous section by themselves. The order in which the participants had to complete these tasks was the same for all (T1,T2,T3,T4). Help was only given by the instructor (who was always present during the test) if a participant could not find a particular feature or was really stuck what to do next. Participants could do the experiment at their own pace.

Finally, participants were asked to give a score to 12 multiple choice questions related to intuitiveness, expressiveness and efficiency:

Q1 How clear did you find it to get an overview of the scenes present? *Hoe duidelijk vindt u het om een overzicht te krijgen van de scenes die aanwezig zijn?*

Q2 How easy did you find it to modify existing scenes? *Hoe makkelijk vindt u het om bestaande scenes te wijzigen?*

Q3 How easy did you find it to create new scenes? *Hoe makkelijk vindt u het om nieuwe scenes te maken?*

Q4 How simple did you find it to set context restrictions for scenes (stage, actor, prop preconditions)? *Hoe eenvoudig vindt u het om context situaties voor scenes (stage, actor, prop beperkingen) in te stellen?*

Q5 How simple did you find it to set actions for scenes? *Hoe eenvoudig vindt u het om acties voor scenes in te stellen?*

Q6 Does the system give you enough freedom to set scene preconditions (stage, actor, prop restrictions)? *Geeft het systeem u genoeg keuzevrijheid om de precondities van scenes (stage, actor, prop beperkingen) in te stellen?*

Q7 Does the system give you enough freedom to set scene actions? *Geeft het systeem u genoeg keuzevrijheid om de acties van scenes in te stellen?*

Q8 Does the system give you enough freedom to set the content of actions? *Geeft het systeem u genoeg keuzevrijheid om de inhoud (content) van de acties van scenes in te stellen?*

Q9 What do you think of the number of steps to modify existing scenes in the simulation? *Wat vindt u van het aantal stappen om bestaande scenes in de simulatie te wijzigen?*

Q10 What do you think of the number of steps to modify existing scenes in the live system? *Wat vindt u van het aantal stappen om bestaande scenes in het live systeem te wijzigen?*

Q11 What do you think of the number of steps to create new scenes in the simulation? *Wat vindt u van het aantal stappen om nieuwe scenes in de simulatie te creëren?*

Q12 What do you think of the number of steps to create new scenes in the live system? *Wat vindt u van het aantal stappen om nieuwe scenes in het live syteem te creëren?*

Each multiple choice question had a scale from 1 to 5. In the intuitiveness category, questions ranged from 1: *very difficult (zeer ingewikkeld* (on the left of the scale) to 5: *very easy (zeer eenvoudig).* In the efficiency category, questions ranged from 1: *very inefficient (zeer onefficient* (on the left of the scale) to 5: *very efficient (zeer efficient).* The questions in the expressiveness category varied between 1: *very insufficient (zeer onvoldoende)* to 5: *very sufficient (zeer voldoende).* Question Q1 was aimed at testing hypothesis H1. Questions Q2, Q3, Q4 and Q5 were asked to verify hypothesis H2, questions Q6 to Q8 to test hypothesis H3 and questions Q9 to Q12 to test hypothesis H4. Next to the multiple choice questions we asked them to give their comments about the intuitiveness, expressiveness and efficiency of the system.

7.1.4 Results

Intuitiveness

The mean scores for the questions in the intuitiveness category are depicted in Figure 7.3: How clear did you find it to get an overview of the scenes present (Q1)? (Mean:3.73,St.Dev.:0.88). How easy did you find it to modify existing scenes (Q2)? (Mean:4.00,St.Dev.:0.53) How easy did you find it to create new scenes (Q3)? (Mean:4.33,St.Dev.0.49) How simple did you find it to set context restrictions for scenes (stage, actor, prop preconditions) (Q4)? (Mean:3.8,St.Dev.:0.94) How simple did you find it to set actions for scenes (Q5)? (Mean:4.4,St.Dev.:0.51)

In terms of ease of use, people seemed to have more difficulty in setting the preconditions of a scene in terms of stage, actor and prop restrictions than with setting the action of a device but found the system intuitive as a whole. This confirms hypothesis H1 and H2. Two participants mentioned in the interview afterwards it was easy to set device actions. One participant commented that the user interface had few options which was something he liked. Aspects that were mentioned that could be improved were the navigation and zoom buttons to center quickly to the shop window area and buttons to open the precondition and action screens. Many participants tried to click on the precondition or action screen in the scene overview screen which did not work. Two participants also remarked that the tool had too much flexibility and suggested a wizard for

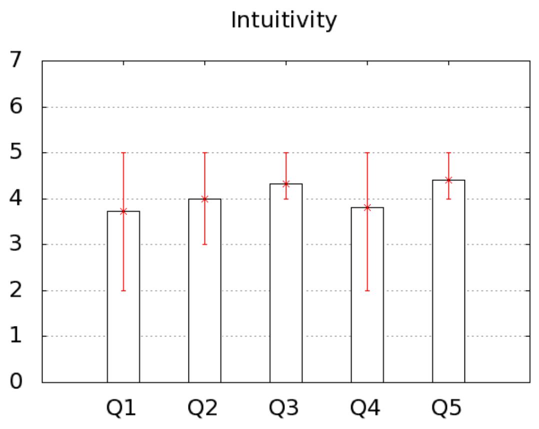


Figure 7.3: Mean scores for questions on ease of use. Horizontal axis shows the five questions asked. Error bars show minimum and maximum scores.

setting preconditions or restrict the user interface to providing only a single way to set device actions (i.e. clicking on a device). Participants recognized that the editor used in the 3D simulation and live system were identical when asked what they found easy and difficult in the live system.

Expressiveness

Figure 7.4 shows the mean scores for the questions in the expressiveness category: Does the system give you enough freedom to set scene preconditions (stage, actor, prop restrictions) (Q6)? (Mean:3.93,St.Dev.:0.46) Does the system give you enough freedom to set scene actions (Q7)? (Mean:3.87,St.Dev.:0.83) Does the system give you enough freedom to set the content of actions (Q8)? (Mean:3.2,St.Dev.:0.94)

Participants were generally satisfied with the possibilities we showed them in terms of setting stage, actor and prop constraints and setting scene actions. This result was surprising because we had expected them to be far less forgiving (and we would have rejected hypothesis H3). The only aspect that was rated considerably less was the possibility to set the content of actions. We had no built-in editor or import function to change the content on the shop window display so people could only choose between presets. Among the many improvements and extra features that could be added (more sensors, more actuators, dynamic light colour effects) this ability to modify the contents (graphics and text) on the window was most mentioned.

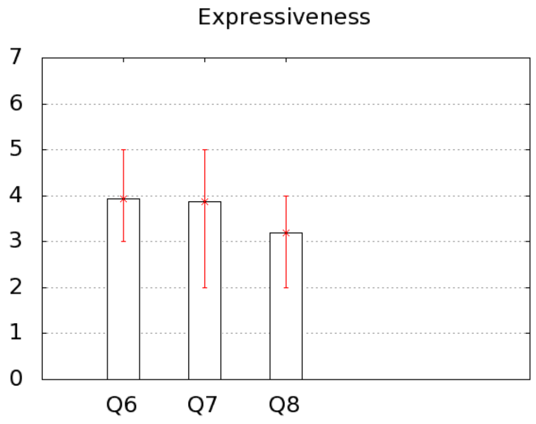


Figure 7.4: Mean scores for questions on expressive power and freedom of choice. Horizontal axis shows the three questions asked. Error bars show minimum and maximum scores.

Efficiency

The mean scores in the efficiency category are depicted in Figure 7.5: What did you think of the number of steps to modify existing scenes in the simulation (Q9)? (Mean:3.40,St.Dev.:0.82) What did you think of the number of steps to create new scenes in the simulation (Q10)? (Mean:3.53,St.Dev.:0.83) What did you think of the number of steps to modify existing scenes in the live system (Q11)? (Mean:3.87,St.Dev.:0.64) What did you think of the number of steps to create new scenes in the live system (Q12)? (Mean:3.87,St.Dev.:0.64)

With respect to efficiency the scores were slightly lower than the scores for intuitiveness but hypothesis H4 is still confirmed (well above average scores). To select the context situation screen from the scene overview screen we placed an icon in the top of the scene overview screen. This icon would maximize the preconditions screen, but all participants tried to click directly on the context area in the scene overview screen to bring up the preconditions screen. Other aspects mentioned afterwards in the interview is that designers would like to set device actions for multiple devices at the same time, and the limitation that there was no undo button. Another comment participants made is that they first had to save the scene before seeing the effect appear. Instead the icons in the context overview screen could show some animation or colour.

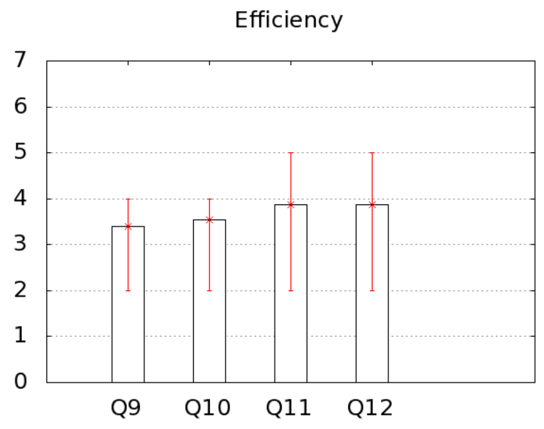


Figure 7.5: Mean scores for questions on efficient use. Horizontal axis shows the four questions asked. Error bars show minimum and maximum scores.

7.2 System performance discussion

For a period of two weeks the prototype was intensively tested on its real-time performance by demonstrating it to the visitors of a yearly internal research exhibition. Like the monolithic system, the modular system responded correctly and in real-time to changes in context situations. However when people caused scenes to switch very quickly (e.g. by standing in between two different interactive zones in front of the shop windows) the ambient narrative engine component could not keep up with the speed of context changes resulting in delays and device actions that would lag behind. The underlying reason was the asynchronous event mechanism that would calculate the new state after each logical context change. The problem can be corrected by implementing a synchronous eventing mechanism in the ambient narrative engine that divides logical events in fixed time slots or frames and recalculates the ambient narrative state at the end or beginning of each frame. Such a mechanism is also needed if we want to check on time or date in the beat preconditions.

The performance of the overall system is dependent on the hardware sensors and embedded software being used. The editor component showed a correct real-time overview of the physical context and did not cause performance problems but measuring errors in the UWB location positioning system could sometimes cause tags to appear and disappear in unexpected places in some cases. The performance of the sensor infrastructure can be improved by monitoring for abnormal sensor events (e.g. using domain knowledge) and duplicating sensors (e.g. extra sensors of the same type but using a different underlying technology). The shop window system used two different location positioning systems to track people in front of the shop window as the UWB signal was reflected by the shop window displays.

The timing and synchronization of device actions caused no problems for the intelligent shop window scenario but could raise issues when video and music should be synchronized over different devices. The rendering engine component does not synchronize device clocks as discussed in [Doornbos, Clout & Ten Kate, 2003].

Finally, in setting up the prototype system we were reminded that detecting bugs in distributed systems can be a difficult and time consuming process. Well-structured software engineering practices and interactive applications to test individual components help to control the complexity.

7.3 Cross-domain application discussion

The expressive power of the ambient narrative intelligent shop window architecture is difficult to evaluate with a user study. Users can only provide feedback on the functionality and interface of the working prototype. Users cannot know if certain functionality is unavailable because it is supported by the system architecture but not implemented in the evaluation system or unsupported by the system architecture (and not implemented). This section starts a discussion about the generality of the prototype system and the limitations that we placed upon ourselves.

7.3.1 Adding new actuators

With the regard to the output side of the system architecture, there are few compromises made by the prototype system. Writing new actuators requires programming skills but once they have been written and conform to the interface and protocol of the rendering engine component and can be detected by the context server, no modification is needed in the system architecture. Once a new actuator has been written and tested, it can be reused as part of many applications and different domains and configured by end-users. The prototype system can therefore easily be extended with different types of luminaires, software applications and all kinds of other output devices.

One of the challenges in writing such actuators is to find the right balance between generality and level of detail. If an actuator is highly generic, there is a risk that it will take the user too much time to configure device actions, if the actuator is very specific, the user may feel restricted. In the intelligent shop window prototype the actuators that control the shop window displays are application specific whereas the luminaire actuators are more generic. By leaving the level of abstraction up to actuator developers, the system architecture can support a wider variety of applications and a larger group of end-users. Novice users may choose more specific actuators that can be configured, expert users will opt for more general actuators than can be programmed.

7.3.2 Adding new sensors

The strongest limitations of the prototype system with respect to generality can be found at the input side and in the partial implementation of the formal ambient narrative model by the ambient narrative engine. The closer we move towards the full ambient narrative model of Chapter 4, the more finegrained the context situations and preconditions that can be scripted and the more diverse applications can be written by end-users. In the prototype system gaze and touch interaction is for example not under control of the ambient narrative engine (and not supported by the editor) and this prevents end-users from declaring a rule that says that when a person is e.g. looking at product X in the shop window, light tile Y will be illuminated.

Adding new sensors is more complicated than adding actuators because it affects the context server, ambient narrative engine and editor components of the system architecture. The context server is affected because it must filter and sense more and other types of physical context data (time, orientation, touch, gaze relations between objects). The ambient narrative engine is affected because this results in new logical context changes and new precondition checks that must be implemented. Finally, the editor must be made aware of the sensor infrastructure and the capabilities of individual sensors for setting preconditions.

The more detailed the preconditions and actions that can be scripted, the more scripts may be needed to implement behavior and the more difficult the authoring task can become for the end-user. If a sensor fails or is removed, the user might still be able to define certain context situations, but the system should give feedback that this situation can never be detected with the current set of sensors.

7.4 Conclusions

In this chapter we investigated the ambient narrative prototype from a usability, performance and extensibility angle.

From the user study we conclude that users found both the 3D simulation and in-situ authoring environment intuitive in use and could create and modify scenes in the prototype system without much difficulty. Overall participants were also satisfied with the possibilities the editor offered to them, but did see room for additional features. This result was somewhat surprising to us as we did not expect participants to accept the limited functionality offered by the editor. In terms of efficiency users rated the prototype slightly lower than in terms of intuitiveness but in general we conclude participants were satisfied with the tool's efficiency. Based on this outcome we therefore confirm the main hypothesis: End-users are able to design ambient narratives using this system.

Through observations of the system in use we draw the conclusion that the system performance meets the real-time response requirement and comparison with a monolithic system but can be improved in a number of areas, most notably the event handling of the ambient narrative engine.

Finally with respect to the system architecture and its extensibility to other types applications and application domains we conclude that the largest compromises to cross domain application are made by the fixed sensor infrastructure assumed by the editor and the partial implementation of the formal ambient narrative model in the prototype system.

8

Conclusions and Future Work

This last chapter first presents a summary of the conclusions reached in the thesis and then discusses open issues and suggestions for future work.

8.1 Conclusions

This thesis addressed the problem of deriving an organizing concept that captures the way in which we form experiences in our everyday life and integrates interactive ambient media into that process. It proposed the design of an end-user software engineering environment that enables end-users, i.e. retailers and designers, to create, deploy and maintain their own smart retail environments described in this general interaction concept to lower the total costs of creating, deploying and maintaining ambient intelligent environments.

In order to derive such a general interaction concept we started in Chapter 2 by analyzing the social-cultural and economical factors that shape the Ambient Intelligence landscape to understand better how ambient intelligence exactly supports people in performing their everyday life activities. We learned that performance and play are not only found in the theatre but can be seen all around us as we perform (often unconsciously) culturally defined social scripts that we have learned through experience. Social interaction itself can be seen as an improvised performance, shaped by the environment and the audience. We discussed the emergence and trends in the experience economy towards multi-sensory co-creation environments and applied literary semiotics theory to describe the interaction between people and the environment in which they perform their social cultural scripts on a more abstract level. This led us to the *ambient narrative* interaction concept as a *spatial-temporal multi-sensory interactive narrative consisting of interrelated media-enhanced social script fragments that are simultaneously read and rewritten by the performance of actor(s) (human or machine)* and *ambient intelligence* as *the part of the emerging story in an ambient narrative as the result of the collective performance of actors that is conveyed through the set of user interfaces of the devices surrounding the actors in the environment.* In addition we provided genre classifications of ambient narratives based on the type of experience and service application domain.

To work towards a concrete system design that can be evaluated with real end-users the decision was made to concentrate on retail and intelligent shop window environments (Chapter 3). Through a series of interviews and workshops conducted with retailers, designers and consultants, where we asked participants to design their own interactive retail experience, we learned that people can decompose user scenarios into individual scenes consisting of a social script and device action(s), giving us the confidence that they could think in the ambient narrative mental model. As part of the interviews we proposed four different strategies to compose scenes and found out that designers had a preference for a 3D simulation scenario in which they could create and test scenes behind a PC in their office, while retailers found programming in-situ using a PDA more appropriate and useful for finetuning previously made scenes. Based on this result we decided to implement both authoring strategies and support both target user groups and improve the workflow between these different stakeholders in one end-user software engineering environment. The user-generated retail ambient narratives and results of a literature study were combined and analyzed for requirements to be placed on an ambient narrative system with authoring support for smart retail environments. This resulted in a list of thirty functional requirements (Table 3.2), grouped into four categories: *ambient narrative concept implementation*, *run-time system performance*, *end-user authoring support* and *system extensibility*. We also presented an intelligent shop window scenario as a carrier application to evaluate the ambient narrative concept in practice.

Based on the ambient narrative concept and run-time system performance functional requirements, we introduced a formal description of the problem of reconstructing ambient intelligence from the modular fragments (beats) in an ambient narrative in Chapter 4. Each beat consists of a preconditions part (social script) and an action part (associated device action(s)). The author of an ambient narrative can add or remove triggers or context-aware queries in the action part to control which beats in the ambient narrative can be activated at a particular moment. We defined a hypertext model to represent beats and made a distinction between a core model, sufficient to implement the intelligent shop window scenario, and a model that could describe all the user-generated ambient narratives we collected. We described how beats are sequenced by an *ambient narrative engine* based on physical context information and session/story state into a coherent story or ambient intelligent experience. First the special case of a static set of beats was discussed followed by a formal description for the more general case of a dynamically changing collection of beats to support experience co-creation environments such as end-user programming environments where end-users can create, modify or delete beats during run-time.

The beat sequencing algorithm implemented by the ambient narrative engine in Chapter 5 essentially consists of three main steps: *updating state*, *action selection* and *action rendering*. After each logical context change or change in story state (*updating state* step), the narrative engine must check which active beats have become invalid and update the active beat set if needed. The narrative engine then has to determine whether new beats need to be activated by checking the preconditions of the currently active triggers. Because the (de)activation of beats can affect the story state and influence the beat sequencing process, a beat sequence iteration only stops when no new beats are activated or existing beats deactivated (*action selection* step). We provided guidelines how to prevent race conditions that might occur. At the end of a beat sequence iteration the active beat set is compared with the previous iteration and the actions of the beats that have changed during an iteration are communicated to the rendering engine that controls the timing and synchronization of device actions over the devices in the intelligent environment (*action rendering* step). The algorithm to modify the beats of an ambient narrative at run-time essentially changes the underlying hypertext graph by adding or removing triggers to control the activation of beats and then calculates which beats need to be (de)activated in a modified version of the beat sequencing algorithm.

The remaining set of functional retail requirements was used to shape the system architecture of an intelligent shop window system that supports the entire lifecycle of ambient narratives. We identified six main components in this end-user software engineering and run-time environment in Chapter 6: *context server*, *ambient narrative engine*, *rendering engine*, *editor*, *simulator* and *exchange server*. The context server maintains a physical context model by aggregating and filtering raw sensor data. It also forwards events to the ambient narrative engine that may cause changes in its logical context model. The ambient narrative engine uses the beat sequencing algorithm outlined above to determine which beats should be rendered next by the rendering engine. The editor component implements the user interface to visualize the state of the intelligent shop window environment and provides the functionality for the designer to modify scenes. The simulator provides the 3D visualization interface to simulate the behavior of ambient narratives. This component reused the same run-time environment as the live system but replaced the real sensors and actuators with virtual ones. This way differences between the live system and simulator can be localized back to the sensors and actuators. The exchange server component offered support for versioning and uploading/downloading ambient narratives to a central repository.

In Chapter 7 we performed a user study to evaluate the usability of the authoring environment, an observation-in-use test to assess the run-time performance of the prototype system and a technical analysis to expose the limitations we placed upon ourselves by restricting ourselves to the intelligent shop window application. In the user study we asked participants with a retail or design background to complete four tasks using the two different authoring environments and assess the system on its intuitiveness, expressiveness and efficiency. In terms of intuitiveness the results showed that people could easily find an overview of the state of the intelligent shop window ambient narrative and had no significant trouble in creating and modifying the behavior of the intelligent shop window ambient narrative with the editor. With respect to the expressive power the current implementation offered, people were satisfied but saw many improvements, in particular the need to be able to modify the text and graphics of the modes that could be chosen for the shop window displays. A similar story held true for the efficiency of the editor. Through observations of the system in use we drew the conclusion that the system performance meets the real-time response requirement and comparison with a monolithic system but can be improved in a number of areas, most notably the event handling of the ambient narrative engine. With respect to the system architecture and its extensibility to other types of applications and application domains we conclude that the largest compromises to cross-domain application are made by the fixed sensor infrastructure assumed by the editor and the partial implementation of the formal ambient narrative model in the prototype system.

Finally, this thesis provides answers on the four research questions stated in Section 1.3.1. The first question we formulated was how we can represent ambient intelligent environments in general in a way that facilitates mass customization. The answer to this question was given in Section 2.3.3 where we defined the ambient narrative concept and showed how people create their own personal mixed reality story or ambient intelligent experience from modular fragments. The second research question dealt with the ability of end-users to understand the ambient narrative concept and the requirements placed by these users on the design of an ambient narrative system that can easily be programmed by end-users. This question was answered by Chapter 3. The third research question of what a system to control the lifecycle of an ambient narrative shop window environment should look like is answered by Chapter 6 that describes and explains the system architecture. This system architecture incorporates the ambient narrative engine component formally described in Chapter 4 and implemented in Chapter 5. The fourth research question raised was whether our end-users would be able to design intelligent shop window applications using such a system. The user study in Section 7.1 provides the answer to this question.

8.2 Suggestions for future work

Many improvements can be made on the ambient narrative concept itself and the implementation of the system architecture and end-user software engineering environment. In the next sections we discuss six different directions for future work:

- ◇ 3D visualization
- ◇ dynamic sensor infrastructure
- ◇ semi-automatic generation of beats
- ◇ end-user debugging
- ◇ ambient narrative concept

8.2.1 3D visualization

Creating a realistic 3D model of an existing store may require a large amount of effort, depending on the size of the store and level of detail desired. 3D modelling software can make this job easier but still requires much manual effort by the designer.

A semi-automatic approach demonstrated by [Van den Hengel, Dick, Thormählen & Ward, 2007] is to generate realistic 3D models of objects from video by sketching the shape of the object over one or more frames of a video and then applying computer vision techniques to interpret and transform these 2D sketches into 3D models. A designer could create a video of the shop first and then generate 3D models of interesting objects in the shop using this technology that can be placed in the virtual shop. 3D models of actuators such as electronic LCD displays, luminaires etc. could also be quickly created this way, but this 3D model does not describe the effect of the actuator on the environment. The effect of the virtual actuator on the virtual environment must therefore still be programmed manually afterwards by a software engineer. One solution is to build a common library of virtual actuator effects that a designer could choose from and associate with a generated 3D model, but it could be interesting to see whether the effect of a real actuator on its surroundings could be automatically extracted and summarized from a video and transformed into a virtual effect in a virtual environment to make the 3D simulation more realistic.

Another area of improvement is the 3D rendering in itself: The more realistic the light rendering in the virtual world, the less need to modify scenes in the live system afterwards because the lighting in the 3D simulation differed from the real world. Further research into computer graphics is needed to enable real-time cinematic quality 3D rendering.

8.2.2 Dynamic sensor infrastructure

In the current system architecture, new actuators who register themselves with the rendering engine and are detected by the context server will automatically become part of the beat sequencing process and can be programmed by end-users provided they implement the plug-in interface of the editor.

The physical sensor infrastructure on the other hand is not so easy to modify in the current prototype (as was discussed in Section 7.3) because knowledge of this sensor infrastructure is maintained and used by the context server, ambient narrative engine and editor. The current system could be improved by centralizing this knowledge in a separate component that is being shared by these components, but this still would not solve the problem of dealing with unknown new types of sensors. If a new sensor would register itself with the context server, the sensor should be able to describe itself, in particular how it can be programmed by end-users, analogous to an actuator that implements an editor plug-in interface that describes its device actions and how they can be configured by end-users. Knowledge representation and reasoning methods could be investigated as used in several context-aware platforms, e.g. see [Devaraju, Hoh & Hartley, 2007] for an overview, to describe the changing capabilities of a dynamically evolving physical sensor infrastructure layer.

Another question is how the capabilities of this changing sensor infrastructure are articulated to the end-user. If there is, for example, no person identification sensor available in the sensor infrastructure, the author of the ambient narrative should be made aware by the user interface that any beat he tries to define that tests on the name of an actor can never be activated. The user interface should therefore be adapted based on the sensor infrastructure model in a similar way as proposed by [Clerckx, Vandervelpen, Luyten & Coninx, 2006] and visualize the sensor infrastructure, as described in [Oppermann, Broll, Capra & Benford, 2006], to give feedback to the user.

The ambient narrative prototype enables an author to copy an ambient narrative from the 3D simulation to the live system but this functionality is restricted at the moment to situations where both the physical geometry and sensor infrastructure of both the source and target location match. It is therefore interesting to investigate how an ambient narrative that is made in one location can be mapped onto another site with a different physical geometry and sensor infrastructure while preserving the behavior of the original ambient narrative as much as possible.

8.2.3 Semi-automatic generation of beats

Modelling the behavior of a relatively simple case like the intelligent shop window scenario requires just a small effort from the user. The level of effort to design a large, complex ambient narrative that models the intelligent behavior of an entire shop floor, for example, will however be much higher. Here we suggest a number of different ways in which beats could be semi-automatically generated to lower the burden on the author.

One possible approach is to replace fixed action scripts with templates which have parameters that are filled in at run-time based on context information or session state, as discussed in Section 4.1.3. This way the author only has to write one template to personalize a media presentation on the shop window based on the name of a user standing in front of it whereas in the current prototype the author would have to manually write beats for each customer individually.

Another direction to address this issue is to develop high-level actuators that themselves generate beats that are inserted in an ambient narrative (and removed). The user would configure such an actuator using the editor as any other actuator but be unaware that this actuator also modifies the ambient narrative. The beats created by this actuator would be invisible in the authoring tool, only the beat that started this high-level actuator would remain visible to the user.

Software agents that modify an ambient narrative may also learn to adapt the social scripts of beats created by human users over time, using machine learning techniques, to make the environment appear more proactive and unpredictable. In general we believe the initiative for this type of adaptation should lie with the users of an intelligent environment and not with the system itself, but there are exceptions like entertainment type of ambient narratives where users are confronted with challenges they have to solve in order to proceed to the next level.

8.2.4 End-user debugging

When ambient narratives grow larger it becomes increasingly difficult for end-users to understand the behavior of an ambient narrative and correct errors, in particular in co-creation situations where both human and software agents dynamically modify beats based on the current state of the ambient narrative.

One improvement over the current editor would be an option for the user to ask the system why a particular scene is (not) active at a particular time. An example of this approach is the Whyline debugging tool described by [Ko & Myers, 2004]. Another direction would be to explore the use of hierarchical layers in ambient narratives and hiding detailed information about the state of the ambient narrative to novice users.

More research on methods and techniques to understand the complexity of their applications and trace back errors in the design is needed to deal with dynamically changing co-creation environments.

8.2.5 Ambient narrative concept

The ambient narrative concept itself is also worth more attention. One possible area of future work is to look at scalability and distributed ambient narratives. There is a maximum to the number of events that can be processed in a single time frame by a single ambient narrative engine. One way to address this scalability issue is to divide a large space up into smaller segments, each covered by one engine. The larger the space covered by a single ambient narrative engine, the more sensors will typically be connected to its context server and the more logical context changes the ambient narrative engine will have to process in turn. This effectively creates a computer grid with a distributed context model and raises the question of how to share narrative state between engines so that users do not know they are switching from one computer to another in the grid.

In setting up the two authoring environments we "accidentally" coupled the output of the 3D simulation to the output of the live system and were able to control the live system through the simulation. Another direction for future work would be to explore different ways of connecting the I/O of simulated and/or live ambient narratives to each other and see what type of applications are enabled.

The final question we raise is of a more philosophical nature. Will the study of computer science into virtual and mixed reality worlds create a new kind of physics science? Experimental physics tests new theories on nature itself to deepen our understanding of how the world around us behaves. By building and experimenting with ever increasingly detailed and more powerful virtual and mixed reality environments, computer science may help to increase our knowledge about nature in a bottom-up constructionist way. Perhaps one day the artificially created realities may have disappeared into the background of our everyday life, so much integrated in our way of living that we simply take them for granted and forget how life was without them.

References

Papers and book chapters

- ◇ M. van Doorn, A.P. de Vries and E. Aarts, “End-user Software Engineering of Smart Retail Environments: The Intelligent Shop Window”, *Proceedings of the Second European Conference on Ambient Intelligence*, Nürnberg, Germany, November 2008.
- ◇ M. van Doorn, E. van Loenen, and A.P. de Vries, “Deconstructing Ambient Intelligence into Ambient Narratives: Intelligent Shop Window”, *Proceedings of the First International Conference on Ambient Media and Systems*, Quebec, Canada, February 2008, pp. 1-8.
- ◇ M. van Doorn, E. van Loenen, and A.P. de Vries, “Performing in Ambient Narratives: Supporting Everyday Life Performances with Technology”, The Drama Review, Volume 51, Number 4, Winter 2007, pp.68-79.
- ◇ E. van Loenen, M. van Doorn, T. Lashina, K. van Gelder, V. Teeven, R. van Haasen, W. de Bruijn, *chapter Interactive Shop Windows in Ambient Lifestyle: From Concept to Experience* (eds. E. Aarts and E. Diederiks), BIS Publishers, 2006, pp. 199-202.
- ◇ M. van Doorn and A.P. de Vries, “Co-creation in Ambient Narratives”, *Ambient Intelligence for Everyday Life*, Lecture Notes in Computer Science 3964, 2006, pp. 103-129.
- ◇ M. van Doorn, R. van Kranenburg, and T. Smalec, “Performance Studies Discussion New York University”, *Contact Quarterly*, Vol.31, No.1, Winter/Spring 2006, pp.50-53.
- ◇ M. van Doorn, “Inside Story on the Experience Economy”, *European Centre for the Experience Economy website*, February 2006 http://www.experience-economy.com/2006/02/20/inside-story-on-the-experience-economy-by-mark-van-doorn-philips-research/
- ◇ M. van Doorn and A.P. de Vries, “Co-creation in Ambient Narratives”, *Proceedings of the Workshop Ambient Intelligence for Everyday Life*, San Sebastian, Spain, July 2005, pp.137-148.

◇ M. van Doorn and A.P. de Vries, “Mass Customization in Ambient Narratives”, *Proceedings of the Philips Conference on Internet Technology*, Eindhoven, The Netherlands, December 2004.

◇ H. ter Horst, M. van Doorn, W. ten Kate, N. Kravtsova and D. Siahaan, “Context-aware Music Selection Using the Semantic Web”, *Proceedings of the 14th Belgium-Netherlands Conference on Artificial Intelligence*, Louvain, Belgium, October 2002, pp. 131-138.

Patent applications

◇ M. van Doorn and E. van Loenen, “Game Logic Portability between Virtual and Real-world Game Engines”, filed 2008.

◇ M. van Doorn, “In-situ End-user Programming of Intelligent Environments Using Smart Tags”, filed 2007.

◇ M. van Doorn, “Device and Method for Controlling a Lighting System by Proximity Setting of a Spotlight Control Device”,WO-2008001277.

◇ M. van Doorn, “Method for Programming by Rehearsal”, WO-2007034455.

◇ M. van Doorn, “A Data Processing System and a Method of Operating a Rendering Platform”, WO-2007026321.

◇ T. Lashina, G. Hollemans, E. van Loenen, S. van de Wijdeven, K. van Gelder, M. van Doorn and V. Buil, “Light Feedback on Physical Object Selection”, WO-2007141675

◇ M. van Doorn, “System and Method for Creating Artificial Atmosphere”, WO-2007069143.

◇ M. van Doorn, “Activity-related Document Management”, WO-200605929.

◇ M. van Doorn, “System, Apparatus and Method for Augmented Reality Glasses for End-user Programming”, EP20060795660.

◇ M. van Doorn, “Controlling Application Devices Simultaneously”, WO-2003058575.

Bibliography

5W!TS [2007], Tomb Experience, Boston, http://www.5-wits.com/.

AARSETH, E. [1997], *Cybertext: Perspectives on Ergodic Literature*, Johns Hopkins University Press.

AARTS, E., R. HARWIG, AND M. SCHUURMANS [2002], Ambient Intelligence, in: P. Denning (ed.), *The Invisible Future*, McGraw-Hill, 235–250.

AARTS, E., AND S. MARZANO (eds.) [2003], *The New Everyday: Views on Ambient Intelligence*, 010 Publishers.

ABOWD, G., C. ATKESON, J. HONG, S. LONG, AND R. KOOPER [1997], Cyberguide: A Mobile Context-aware Tour Guide, *Wireless Networks* **3**, 421–433.

ABRAS, C., D. MALONEY-KRICHMAR, AND J. PREECE [2004], User-Centered Design, in: W. Bainbridge (ed.), *Encyclopedia of Human-Computer Interaction*, Berkshire Publishing Group.

ADOBE [2008], Adobe Flash Professional, http://www.adobe.com/.

APPLE [2008], iDVD, http://www.apple.com/ilife/idvd/.

ARGNET [2008], Alternate Reality Gaming Network, http://www.argn.com/.

AUTODESK [2008], 3D Studio Max, website, http://www.autodesk.com.

AZUMA, R. [1999], *Mixed Reality: Merging Real and Virtual Worlds*, Chapter The Challenge of Making Augmented Reality Work Outdoors, 373–390. Springer-Verlag.

BALLAGAS, R., F. MEMON, R. REINERS, AND J. BORCHERS [2007], iStuff Mobile: Rapidly Prototyping New Mobile Phone Interfaces for Ubiquitous Computing, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, San Jose, USA, 1107–1116.

BARON, P. [2006], Location-based Mobile Phone Games, www.in-duce.net/archives/locationbased_mobile_phone_games.php.

BARRENHO, F., T. ROMÃO, T. MARTINS, AND N. CORREIA [2006], InAuthoring Environment: Interfaces for Creating Spatial Stories and Gaming Activities, *Proceedings of the ACM International Conference on Advances in Computer Entertainment Rechnology*, Hollywood, California, 9.

BARTON, J., AND V. VIJAYARAGHAVAN [2003], UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment, HP Technical Report, http://www.hpl.hp.com/techreports/2003/HPL-2003-93.html.

BATES, J., A. LOYALL, AND W. REILLY [1991], Broad Agents, *Sigart Bulletin* **2**, 38–40.

BAUDRILLARD, J. [1985], *Distinction: a Social Critique of the Judgment of Taste*, translated by Richard Nice, Harvard University Press.

BAUDRILLARD, J. [1995], *Simulacra and Simulation*, University of Michigan Press.

BEISER, V. [2006], Bagdad, USA, *Wired* **14**, http://www.wired.com/wired/archive/14.06/iraq.html.

BELL, M., M. CHALMERS, L. BARKHUUS, M. HALL, AND S. SHERWOOD [2006], Interweaving mobile games with everyday life, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Montreal, Quebec, Canada, 417–426.

BIERZ, T. [2006], Interaction Technologies for Large Displays - An Overview, in: H. Hagen, A. Kerren, and P. Dannenmann (eds.), *Visualization of Large and Unstructured Data Sets*, Lecture Notes in Informatics 4, Springer.

BINSUBAIH, A., S. MADDOCK, AND D. ROMANO [2005], Game Logic Portability, *Proceedings of the International ACM Conference on Advances in Computer Entertainment Technology*, Valencia, Spain, ACM, 458–461.

BITNER, J. [1992], Servicescapes: The Impact of Physical Surroundings on Customers and Employees, *Journal of Marketing* **54**, 69–82.

BJÖRK, S., J. HOLOPAINEN, P. LJUNGSTRAND, AND K. AKESSON [2002], Designing Ubiquitous Computing Games A Report from a Workshop Exploring Ubiquitous Computing Entertainment, *Personal Ubiquitous Computing* **6**, 443–458.

BLACKWELL, A., AND R. HAGUE [2001], AutoHAN: An Architecture for Programming the Home, *Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)*, 150–157.

BLEEKER, J. [2006], A Manifesto for Networked Objects Cohabiting with Pigeons, Arphids and Aibos in the Internet of Things, http://www.nearfuturelaboratory.com/files/WhyThingsMatter.pdf.

BLOW, J. [2004], Game Development: Harder Than You Think, *ACM Queue* **1**, pp. 28–37.

BOEHM, B. [1988], A Spiral Model of Software Development and Enhancement, *IEEE Computer* **21**, 61–72.

BOJE, D., T. ADLER, AND J. BLACK [2005], Theatrical Façades and Agents in a Synthesized Analysis from Enron Theatre: Implications to Transaction Cost and Agency Theories, *Tamara: Journal of Critical Postmodern Organization Science* **3**, 39–56.

BOJE, D., J. LUHMAN, AND A. CUNLIFFE [2003], A Dialectic Perspective on the Organization Theatre Metaphor, *American Communication Journal* **6**, 1–16, http://acjournal.org/holdings/vol6/iss2/articles/boje.htm.

BOSWIJK, A., T. THIJSSEN, AND E. PEELEN [2007], *The Experience Economy: A New Perspective*, Pearson Education.

BOWEN, J., AND S. FILIPPINI FANTONI [2004], Personalization and the Web from a Museum Perspective, *Proceedings of Museums and the Web*, Archives & Museum Informatics.

BRAND, S. [1995], *How Buildings Learn: What Happens After They're Built*, Penguin.

BRUSILOVSKY, P. [1996], Methods and Techniques of Adaptive Hypermedia, *User Modeling and User-Adapted Interaction* **6**, 87–129.

BURKE, K. [1966], *Language as Symbolic Action: Essays on Life, Literature and Method*, Berkeley: University of California Press.

BURNETT, M., C. COOK, AND G. ROTHERMEL [2004], End-user software engineering, *Communications of the ACM* **47**, 53–58.

BURNETT, M., G. ENGELS, B. MYERS, AND G. ROTHERMEL (eds.) [2007], *End-User Software Engineering, 18.02. - 23.02.2007*, Dagstuhl Seminar Proceedings 07081, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

CAMERON, A. [1995], Dissimulations: The Illusion of Interactivity, *Millenium Film Journal* **28**, 33–47.

CAO, H., P. OLIVIER, AND D. JACKSON [2008], Enhancing Privacy in Public Spaces Through Crossmodal Displays, *Social Science Computer Review* **26**, pp. 87–102.

CARROLL, J. (ed.) [1995], *Scenario-Based Design: Envisioning Work and Technology in System Development*, John Wiley & Sons, Inc., New York, NY, USA.

CARSON, D. [2000], Environmental Storytelling: Creating Immersive 3D Worlds Using Lessons Learned From the Theme Park Industry, http://www.gamasutra.com/features/20000301/carson_pfv.htm.

CAVAZZA, M., J. LUGRIN, D. PIZZI, AND F. CHARLES [2007], Madame Bovary on the Holodeck: Immersive Interactive Storytelling, *Proceedings of the 15th ACM Conference on Multimedia*, Augsburg, Germany, 651–660.

CHANDLER, D. [2001], *Semiotics: The Basics*, Routledge.

CHATMAN, S. [1978], *Story and Discourse: Narrative Structure in Fiction and Film*, Ithaca: Cornell University Press.

CHATWIN, B. [1988], *The Songlines*, Pinguin Books.

CHEN, H., F. PERICH, AND D. CHAKRABORTY [2004], Intelligent Agents Meet Semantic Web in a Smart Meeting Room, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, Washington, DC, USA, IEEE Computer Society, 854–861.

CHEOK, A., K. GOH, WEI LIU, F. FARBIZ, S. FONG, AND S. TEO [2004], Human Pacman: A Mobile, Wide-area Entertainment System based on Physical, Social, and Ubiquitous Computing, *Personal Ubiquitous Computing* **8**, 71–81.

CHEVERST, K., N. DAVIES, K. MITCHELL, AND A. FRIDAY [2000], Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, The Hague, The Netherlands, 17–24.

CHING, F. [1996], *Architecture: Form, Space and Order 2nd edition*, Wiley.

CHU, T-Y. LIU; T-H. TAN; Y-L. [2006], The Ubiquitous Museum Learning Environment: Concept, Design, Implementation, and a Case Study, *Proceedings of the Sixth IEEE Conference on Advanced Learning Technologies*, 989–991.

CLARK, T., AND I. MANGHAM [2004], From Dramaturgy to Theatre as Technology: The Case of Corporate Theatre, *Journal of Management Studies* **41**, pp. 37–59.

CLERCKX, T., C. VANDERVELPEN, K. LUYTEN, AND K. CONINX [2006], A Task-driven User Interface Architecture for Ambient Intelligent Environments, *Proceedings of the 11th ACM Conference on Intelligent User Interfaces*, Sydney, Australia, 309–311.

CMIL [2008], Contextual Media Integration Language (CML), http://www.oacea.com/cmil/.

COLORKINETICS [2008], Philips Solid State Lighting Systems, http://www.colorkinetics.com/.

CRYSTALSPACE [2008], CrystalSpace Game Engine, website, http://www.crystalspace3d.org.

CSERKUTI, P., T. LEVENDOVSZKY, AND H. CHARAF [2006], Survey on Subtree Matching, *Proceedings of the International Conference on Intelligent Engineering Systems*, 39, 216–221.

CSIKSZENTMIHALYI, M. [1990], *Flow: The Psychology of Optimal Experi-*

ence, New York: Harper and Row.

CYPHER, A. [1993], *Watch What I Do: Programming by Demonstration*, MIT Press.

DATABASE OF VIRTUAL ART [2008], Database of Virtual Art, http://www.virtualart.at.

DAVENPORT, G., AND M. MURTAUGH [1997], Automatist Storyteller Systems and the Shifting Sands of Story , *IBM Systems Journal* **36**, 446–456.

DAVIS, S. [1987], *Future Perfect*, Reading.

DE BRA, P., G-J. HOUBEN, AND H. WU [1999], AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia*, Darmstadt, Germany, 147–156.

DE OLIVEIRA, N., N. OXLEY, AND M. PETRY [2004], *Installation art in the New Millennium: Empire of the Senses*, London: Thames & Hudson.

DERTOUZOS, M. [1999], The Future of Computing, *Scientific American* **281**, 52–55.

DEVARAJU, ANUSURIYA, SIMON HOH, AND MICHAEL HARTLEY [2007], A Context Gathering Framework for Context-aware Mobile Solutions, *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems*, Singapore, 39–46.

DEVMASTER [2008], DevMaster's Game and Graphics Engines Database, http://www.devmaster.net/engines/.

DEY, A., R. HAMID, AND C. BECKMANN [2004], a CAPpella: Programming by Demonstration of Context-aware Applications, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, 33–40.

DIDIER, J., O. HESS, AND M. LUTYENS [2007], House-Swarming, Art Center College of Design, Pasadena, http://www4.alzado.net/edintro.html.

DIEDERIKS, E., AND H. HOONHOUT [2007], Radical Innovation and End-user Involvement: The Ambilight Case, *Journal of Knowledge, Technology & Policy* **20**, 31–38.

DNP [2008], DNP Holo Screen, http://www.en.dnp.dk/get/472.html.

DOORN, M. VAN, AND A.P. DE VRIES [2006], Co-creation in Ambient Narratives, *Ambient Intelligence for Everyday Life, Lecture Notes in Computer Science 3964*, Springer-Verlag, 103–129.

DOORNBOS, R., R. CLOUT, AND W. TEN KATE [2003], *Media Processing Abstraction Layer (MAL) Protocol Specification*, Technical Report PR-TN-2003/00477, Philips Research.

DOW, S., M. MEHTA, E. HARMON, B. MACINTYRE, AND M. MATEAS [2007], Presence and Engagement in an Interactive Drama, *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, San Jose, California, USA, ACM, 1475–1484.

DRAPER, S., D. NORMAN, AND C. LEWIS [1986], Introduction, in: D. A. Norman and S. W. Draper (eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction*, Erlbaum, 1–6.

ECLIPSE [2008], Eclipse: An Open Development Platform, http://www.eclipse.org.

EDVARDSSON, B., B. ENQUIST, AND R. JOHNSTON [2005], Cocreating Customer value through Hyperreality, *Journal of Service Research* **8**, pp. 149–161.

ELIASSON, O. [2003], Weather Project, Tate Modern, London, http://www.tate.org.uk/modern/exhibitions/eliasson/default.htm.

ERICKSON, T. [1996], The World Wide Web as Social Hypertext, *"Viewpoints" in Communications of the ACM* **39**, 15–17.

EVES, D., R. COLE, AND D. CALLAWAY [2008], Ambient Environment Effects, WO/2008/007293.

EXIST [2008], eXist Open Source Native XML Database, http://exist.sourceforge.net/.

FINZER, W., AND L. GOULD [1993], *Rehearsal world: Programming by Rehearsal*, 79–100, Cambridge, MA, USA, MIT Press.

FISHER, W. [1989], *Human Communication as Narration: Toward a Philosophy of Reason, Value, and Action*, Columbia: University of South Carolina Press.

FISK, R., AND S. GROVE [1992], The Service Experience as Theater, *Advances in Consumer Research* **19**, 455–461.

FISK, R., AND P. TANSUHAJ [1985], *Services Marketing: An Annotated Bibliography*, American Marketing Association.

FOGARTY, J., J. FORLIZZI, AND S. HUDSON [2001], Aesthetic Information Collages: Generating Decorative displays that Contain Information, *Proceedings of the 14th ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 141–150.

FOWLER, M. [2003], *UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition)*, Addison-Wesley.

FUJINAMI, K., T. YAMABE, AND T. NAKAJIMA [2004], "Take Me With You!": A Case Study of a Context-aware Application Integrating Cyber and Physical Spaces, *Proceedings of the ACM Symposium on Applied Computing*, Nicosia, Cyprus, 1607–1614.

GALYEAN, T. [1995], *Narrative Guidance of Interactivity*, Ph.D. thesis, MIT

Media Lab.

GARDNER, W., AND B. AVOLIO [1998], The Charismatic relationship: A Dramaturgical Perspective, *Academy of Management Review* **23**, 32–58.

GEPHNER, D., J. SIMONIN, AND N. CARBONELL [2007], Gaze as a Supplementary Modality for Interacting with Ambient Intelligence Environments, *Universal Access in Human-Computer Interaction. Ambient Interaction*, Lecture Notes in Computer Science 4555, Springer.

GLYNN, R. [2008], Interactive Architecture Weblog, http://www.interactivearchitecture.org.

GOβMANN, J., AND M. SPECHT [2002], Location models for augmented environments, *Personal Ubiquitous Computiung* **6**, 334–340.

GOFFMAN, E. [1959], *The Presentation of Self in Everyday Life*, Doubleday: Garden City.

GOTTDIENER, M. [2001], *The Theming of America*, Westview Press.

GRAU, O. [2003], *Virtual Art: From Illusion to Immersion*, MIT Press.

GRØNBÆK, K., PETER ØRBÆK, J. KRISTENSEN, AND M. ERIKSEN [2003], Physical Hypermedia: organising collections of mixed physical and digital material, *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, Nottingham, UK, 10–19.

GRØNBÆK, K., P. VESTERGAARD, AND P. ØRBÆK [2002], Towards Geo-Spatial Hypermedia: Concepts and Prototype Implementation, *Proceedings of the 13th Conference on Hypertext and Hypermedia*, Maryland, USA.

GROVE, S.J., AND R.P. FISK [1983], *Emerging Perspectives on Services Marketing*, Chapter The Dramaturgy of Services Exchange: An Analytical Framework for Services Marketing, 45–9. American Marketing Association.

GROVE, S., R. FISK, AND M. LAFORGE [2004], Developing the Impression Management Skills of the Service Worker: An Application of Stanislavskys Principles in a Services Context, *The Service Industries Journal* **24**, 1–14.

GÜVEN, S., AND S. FEINER [2003], Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality, *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, Washington, DC, USA, IEEE Computer Society, 118.

HALASZ, F., AND M. SCHWARTZ [1994], The Dexter Hypertext Reference Model, *Communications of the ACM* **37**, 30–39.

HANSEN, F., N. BOUVIN, B. CHRISTENSEN, AND K. GRØNBÆK [2004], Integrating the Web and the World: Contextual Trails on the

Move, *Proceedings of the 15th ACM Conference on Hypertext and Hypermedia*, Santa Cruz, USA, 98–107.

HARDMAN, L., D. BULTERMAN, AND G. VAN ROSSUM [1994], The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model, *Communications of the ACM* **37**, 50–64.

HARDMAN, L., Z. OBRENOVIC, F. NACK, B. KERHERVÉ, AND K. PIERSOL [2008], Canonical Processes of Semantically Annotated Media Production, *Multimedia Systems* **14**, 327–340.

HARRIS, R., K. HARRIS, AND S. BARON [2003], Theatrical Service Experiences: Dramatic Script Deployment with Employees, *International Journal of Service Industry Management* **14**, 184–199.

HARTMANN, B., L. ABDULLA, AND M. MITTAL [2007], Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, San Jose, USA, 145–154.

HENGEL, A. VAN DEN, A. DICK, T. THORMÄHLEN, AND B. WARD [2007], VideoTrace: Rapid Interactive Scene Modelling From Video, *SIGGRAPH '07*, San Diego, USA, ACM, 86.

HEUVEL, H. VAN DE [2007], Context Awareness in Ambient Intelligence Surroundings, Master thesis, Technical University of Eindhoven.

HORN, M., AND R. JACOB [2006], Tangible Programming in the Classroom: A Practical Approach, *Extended Abstracts of the 23rd ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, 869–874.

HORST, H. TER, M. VAN DOORN, N. KRAVTSOVA, AND W. TEN KATE [2002], Context-aware Music Selection Using Knowledge on the Semantic Web, *Proceedings of the 14th Belgium-Netherlands Conference on Artificial Intelligence*, 131–138.

HTML [2008], HTML 4.01 Specification, http://www.w3.org/TR/html4/.

HUMBLE, J., A. CRABTREE, AND T. HEMMINGS [2003], "Playing with the Bits" User-Configuration of Ubiquitous Domestic Environments, *Proceedings of the 5th IEEE Conference on Ubiquitous Computing*, 256–263.

HYDROPOLIS [2008], Hydropolis Underwater Hotel, http://www.designbuild-network.com/projects/Hydropolis/.

IBM RETAIL [2008], How Immersive Technology Can Revitalize the Shopping Experience, Technical Report, http://www-03.ibm.com/industries/retail/index.jsp.

IZADI, S., M. FRASER, S. BENFORD, AND M. FLINTHAM [2002], Citywide: Supporting Interactive Digital Experiences Across Physical

Space, *Personal Ubiquitous Computing* **6**, 290–298.

JOHN, J., S. GROVE, AND R. FISK [2006], Improvisation in Service Performances: Lessons from Jazz, *Managing Service Quality* **16**, 247–268.

KANG, H., B. BEDERSON, AND B. SUH [2007], Capture, Annotate, Browse, Find, Share: Novel Interfaces for Personal Photo Management, *International Journal of Human-Computer Interaction* **23**, 315–337.

KELLEHER, C., AND R. PAUSCH [2005], Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers, *ACM Computer Surveys* **37**, 83–137.

KESSELS, A. [2006], Intelligent Shop Window: Interaction Styles and Feedback Mechanisms, Master thesis, Technical University of Eindhoven.

KINDBERG, T., J. BARTON, AND J. MORGAN [2002], People, Places, Things: Web Presence for the Real World, *Mobile Network Applications* **7**, 365–376.

KLINGMANN, A. [2007], *Brandscapes: Architecture in the Experience Economy*, MIT Press.

KO, A., AND B. MYERS [2004], Designing the whyline: A debugging interface for asking questions about program behavior, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Vienna, Austria, 151–158.

KORPIPÄÄ, P., E. MALM, I. SALMINEN, AND T. RANTAKOKKO [2005], Context Management for End User Development of Context-aware Applications, *Proceedings of the 6th ACM Conference on Mobile Data Management*, Ayia Napa, Cyprus, 304–308.

KOSAK, D. [2005], Will Wright Presents Spore... and a New Way to Think About Games,
http://www.gamespy.com/articles/595/595975p1.html.

KOZINETS, R., J. SHERRY, B. DEBERRY-SPENCE, AND A. DUHACHECK [2002], Themed Flagship Brand Stores in the New Millenium: Theory, Practice and Prospects, *Journal of Retailing* **78**, 17–29.

KROGH, P. [2000], Interactive rooms - augmented reality in an architectural perspective, *Designing Augmented Reality Environments*, 135–137.

LAUREL, B. [1986], *Toward the Design of a Computer-Based Interactive Fantasy System*, Ph.D. thesis, Ohio State University.

LAUREL, B. [1993], *Computers as Theatre*, Addison-Wesley.

LAZEROMS, M. [2002], Intelligent Personal Care Environment - The Bathroom Cares for You, http://www.research.philips.com/technologies (Mirror Display).

LE CORBUSIER [1960], *Towards a New Architecture*, Vitruvius.

LEGO [2008], Lego Factory, http://factory.lego.com/.

LEGRADY, G. [2000-2002], Sensing Speaking Space, Museum of Modern Art, San Francisco,
http://channel.creative-capital.org/project_78.html.

LI, Y., J. HONG, AND J. LANDAY [2004], Topiary: A Tool for Prototyping Location-enhanced Applications, *Proceedings of the 17th ACM Symposium on User Interface Software and Technology*, Santa Fe, USA, 217–226.

LIEBERMAN, H. [2001], *Your Wish Is My Command: Programming by Example*, Morgan Kaufman.

LINDEN LABS [2008], Second Life: Your World, Your Imagination, http://secondlife.com.

LOENEN, E. VAN, T. LASHINA, AND M. VAN DOORN [2006], *Ambient Lifestyle: From Concept to Experience*, Chapter Interactive Shop Windows, 199–202. Amsterdam:Bis Publishers.

LOHR, S. [2006], Apple, a Success at Stores, Bets Big on Fifth Avenue, *New York Times*, May 19.

LOZANO-HEMMER, R. [2004], Vectorial Elevation, Sky Above Dublin,
http://www4.alzado.net/edintro.html.

MAES, P. [1989], How To Do The Right Thing, *Connection Science Journal* **1**, 291–321.

MAGERKO, B. [2002], A Proposal for an Interactive Drama Architecture, *Artificial Intelligence and Interactive Entertainment*, 76–81.

MARTIN, W. [1986], *Recent Theories of Narrative*, Cornell University Press.

MATEAS, M., AND A. STERN [2003], Façade: An Experiment in Building a Fully-Realized Interactive Drama, *Game Developer's Conference: Game Design Track*, San Jose, California.

MAYHEW, D. [1999], *The Usability Engineering Lifecycle: A Practioner's Handbook for User Interface Design*, Morgan Kaufmann.

MCGONIGAL, J. [2003], This Is Not a Game: Immersive Aesthetics and Collective Play, *Proceedings of the Digital Arts & Culture Conference*, 110–118.

MCHUGH, J. [2006], The Great Escape, *Wired* **14**,
http://www.wired.com/wired/archive/14.03/lafuga.html.

MCKEE, R. [1997], *Story: Substance, Structure, Style and The Principles of Screenwriting*, Regan Books.

MCNERNEY, T. [2004], From Turtles to Tangible Programming Bricks: Explorations in Physical Language Design, *Personal Ubiquitous Computer* **8**, 326–337.

MEADOWS, M. [2002], *Pause and Effect: The Art of Interactive Narrative*, New Riders Publishers.

MICROSOFT [2008], Visual Studio, http://msdn.microsoft.com/en-us/vs2008/default.aspx.

MILLARD, D., D. DE ROURE, D. MICHAELIDES, AND D. THOMPSON [2004], Navigational Hypertext Models For Physical Hypermedia Environments, *Proceedings of the 15th ACM conference on Hypertext and Hypermedia*, Santa Cruz, USA, 110–111.

MONTEMAYOR, J., A. DRUIN, AND G. CHIPMAN [2004], Tools for Children to Create Physical Interactive Storyrooms, *Computers in Entertainment* **2**, 12–12.

MOTT, B., AND J. LESTER [2006], U-director: A Decision-theoretic Narrative Planning Architecture for Storytelling Environments, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, USA, 977–984.

MURRAY, J. [1998], *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*, MIT Press.

NA, J., AND R. FURUTA [2001], Dynamic Documents: Authoring, Browsing, and Analysis using a High-level Petri-net-based Hypermedia System, *Proceedings of the ACM Symposium on Document Engineering*, New York, NY, USA, 38–47.

NARDI, B. [1993], *A Small Matter of Programming: Perspectives on End User Computing*, MIT Press.

NAVAB, N. [2003], Industrial Augmented Reality: Challenges in Design and Commercialization of Killer Apps, *Proceedings of the Second IEEE/ACM Symposium on Mixed and Augmented Reality*, IEEE/ACM, 2–6.

NEWT GAMES [2003], Mogi, http://www.mogimogi.com/mogi.php?language=en.

NICKLAS, D., AND B. MITSCHANG [2001], The NEXUS Augmented World Model: An Extensible Approach for Mobile, Spatially-Aware Applications, *Proceedings of the 7th IEEE Conference on Object-Oriented Information Systems*, 392–401.

NORTON, M., AND B. MACINTYRE [2005], Butterfly Effect: An Augmented Reality Puzzle Game, *Proceedings of the 4th IEEE Symposium on Mixed and Augmented Reality*, 212–213.

O'NEILL, E., D. WOODGATE, AND V. KOSTAKOS [2004], Easing the Wait in the Emergency Room: Building a Theory of Public Information Systems, *Proceedings of the 5th ACM Conference on Designing Interactive Systems*, Cambridge, USA, 17–25.

OPPERMANN, L., G. BROLL, M. CAPRA, AND S. BENFORD [2006], Extending Authoring Tools for Location-Aware Applications with an Infrastructure Visualization Layer, *Ubicomp*, 52–68.

OSWICK, C., T. KEENOY, AND D. GRANT [2001], Dramatizing and Organizing: Acting and Being, *Journal of Organizational Change Management* **14**, pp. 218–224.

OWL [2008], Web Ontology Language (OWL), http://www.w3.org/2004/OWL/.

PELTONEN, P., E. KURVINEN, A. SALOVAARA, AND G. JACUCCI [2008], It's Mine, Don't Touch!: Interactions at a Large Multi-touch Display in a City Centre, *Proceeding of the 26th ACM Conference on Human Factors in Computing Systems*, Florence, Italy, 1285–1294.

PELTONEN, P., A. SALOVAARA, G. JACUCCI, AND T. ILMONEN [2007], Extending Large-scale Event Participation with User-created Mobile Media on a Public Display, *Proceedings of the 6th ACM Conference on Mobile and Ubiquitous Multimedia*, Oulu, Finland, 131–138.

PERLIN, K., AND A. GOLDBERG [1996], Improv: A System for Scripting Interactive Actors in Virtual Worlds, *Proceedings of the 23rd ACM Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 205–216.

PESCOVITZ, D. [2005], Crappy restaurant, BoingBoing, http://www.boingboing.net/2005/06/30/crappy-restaurant.html.

PHILIPS [2007], Wake-up Light, http://www.wakeuplight.philips.com/.

PHYSORG [2007], Worlds First Ambient Experience Suite Opens, http://www.physorg.com/news5443.html.

PINE, J. [1992], *Mass Customization: The New Frontier in Business Competition*, Harvard Business School Press.

PINE, J., AND J. GILLMORE [1999], *The Experience Economy*, Harvard Business School Press.

PINE, J., AND J. GILLMORE [2007], *Authenticity: What Customers Really Want*, Harvard Business School Press.

PINHANEZ, C., K. MASE, AND A. BOBICK [1997], Interval Scripts: A Design Paradigm for Story-based Interactive Systems, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, ACM Press, 287–294.

PRAHALAD, C.K., AND V. RAMASWAMY [2004], *The Future of Competition: Co-creating Unique Value With Customers*, Harvard Business School Press.

PRASOV, Z., AND J. CHAI [2006], Predicting User Attention using Eye Gaze in Conversational Interfaces, *The Third Midwest Computational Linguistics Colloquium*, Urbana-Champaign, IL.

QVARFORDT, P., AND S. ZHAI [2005], Conversing with the user based on eye-gaze patterns, *Proceedings of the 22th ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, 221–230.

RANGANATHAN, A., J. AL-MUHTADI, AND S. CHETAN [2004], MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications, *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, Toronto, Canada, Springer-Verlag, 397–416.

RASHID, O., W. BAMFORD, P. COULTON, AND R. EDWARDS [2006], PAC-LAN: Mixed-reality Gaming with RFID-enabled Mobile Phones, *Computer Entertainment* **4**, 4–21.

RDF [2008], Resource Description Framework (RDF), http://www.w3.org/RDF/.

REEVES, B., AND C. NASS [1996], *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*, Cambridge University Press.

REPENNING, A., AND A. IOANNIDOU [2006], Mobility Agents: Guiding and Tracking Public Transportation Users, *Proceedings of the ACM Working Conference on Advanced Visual Interfaces*, Venezia, Italy, 127–134.

REPENNING, A., AND T. SUMNER [1994], Programming as Problem Solving: A Participatory Theater Approach, *Proceedings of the ACM Workshop on Advanced Visual Interfaces*, Bari, Italy, 182–191.

RESNICK, M., B. MYERS, K. NAKAKOJI, AND B. SHNEIDERMAN [2005], Design Principles for Tools to Support Creative Thinking, *Workshop on Creativity Support Tools*, http://www.cs.umd.edu/hcil/CST/index.html.

RIFKIN, J. [2001], *The Age of Access: The New Culture of Hypercapitalism, Where all of Life is a Paid-For Experience*, J.P. Tarcher.

RIPER, T. VAN [2007], World's 10 Largest Shopping Malls, *Forbes Magazine*, January 9.

RIVKIN, J., AND M. RYAN [1998], *Literary Theory: An Anthology*, Blackwell Publishers.

RODDEN, T., A. CRABTREE, AND T. HEMMINGS [2004], Between the Dazzle of a New Building and its Eventual Corpse: Assembling the Ubiquitous Home, *Proceedings of the 5th ACM Conference on Designing Interactive Systems*, Cambridge, USA, 71–80.

ROMERO, L., AND N. CORREIA [2003], HyperReal: A Hypermedia Model for Mixed Reality, *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, Nottingham, UK, 2–9.

SANDOR, C., B. BELL, AND A. OLWAL [2004], Visual End User Configuration of Hybrid User Interfaces, *Proceedings of the ACM SIGMM*

Workshop on Effective telepresence, New York, NY, USA, 67–68.

SAWYER, K. [2001], The Improvisational Performance of Everyday Life, *Journal of Mundane Behavior* **2**, 149–163.

SCHECHNER, R. [2002], *Performance Studies: An Introduction*, Routledge: New York.

SCHULZE, G. [1992], *Die Erlebnisgesellschaft: Kultursoziologie der Gegenwart*, Campus.

SHERRY, J. [1998], *Servicescapes: The Concept of Place in Contemporary Markets*, Ntc Business Books.

SHIREHJINI, A., AND F. KLAR [2005], 3DSim: Rapid Prototyping Ambient Intelligence, *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence*, New York, NY, USA, 303–307.

SINCLAIR, P., K. MARTINEZ, D. MILLARD, AND M. WEAL [2002], Links in the Palm of your Hand: Tangible Hypermedia using Augmented Reality, *Proceedings of the 13th ACM Conference on Hypertext and Hypermedia*, Maryland, USA, 127 – 136.

SMARTEYE [2008], SmartEye Gaze Tracking Technology, http://www.smarteye.se.

SMIL [2008], Synchronized Multimedia Integration Language (SMIL), http://www.w3.org/AudioVideo/.

SMITH, M., D. DAVENPORT, H. HWA, AND T. COMBS-TURNER [2004], Object Auras: A Mobile Retail and Product Annotation System, *Proceedings of the 5th ACM Conference on Electronic Commerce*, 240–241.

SOHN, T., AND A. DEY [2003], iCAP: An Informal Tool for Interactive Prototyping of Context-aware Applications, *Extended Abstracts ACM Conference on Human Factors in Computing Systems*, Ft. Lauderdale, USA, 974–975.

SOMMERER, C., AND L. MIGNONNEAU [2001], The Living Room, Living in the Future symposium, Malmo, http://www.interface.ufg.ac.at/christa-laurent/WORKS/index.html.

SPARACINO, F. [2003], Sto(ry)chastics: A Bayesian Network Architecture for User Modeling and Computational Storytelling for Interactive Spaces, *Proceedings of the 5th IEEE Conference on Ubiquitous Computing*, New York, NY, USA, 54–72.

STOTTS, P., AND R. FURUTA [1989], Petri-net-based Hypertext: Document Structure with Browsing Semantics, *ACM Transactions on Information Systems* **7**, 3–29.

STROSS, R. [2007], Apple's Lesson for Sony's Stores: Just Connect, http://www.nytimes.com/2007/05/27/business/yourmoney/27digi.html.

STUART, F., AND S. TAX [2004], Toward an integrative approach to designing service experiences: Lessons learned from the theatre, *Journal of Operation Management* **22**, 609–627.

SZILAS, N. [2003], IDtension: A Narrative Engine for Interactive Drama, *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Darmstadt, Germany, Frauenhofer IRB Verlag, 187–203.

SZULBORSKI, D. [2005], *This Is Not A Game: A Guide to Alternate Reality Gaming*, Lulu.

TANG, A., M. FINKE, AND M. BLACKSTOCK [2008], Designing for Bystanders: Reflections on Building a Public Digital Forum, *Proceeding of the 26th ACM Conference on Human Factors in Computing Systems*, Florence, Italy, 879–882.

TEKSCAN [2008], Tekscan Pressure Sensor Technology, http://www.tekscan.com.

THOMAS, B., B. CLOSE, J. DONOGHUE, J. SQUIRES, AND P. DE BONDI [2002], First Person Indoor/Outdoor Augmented Reality Application: ARQuake, *Personal Ubiquitous Computing* **6**, 75–86.

THOMAS, B., W. PIEKARSKI, AND B. GUNTHER [1999], Using Augmented Reality to Visualise Architecture Designs in an Outdoor Environment, *Design Computing on the Net*.

TOLER, L. [2007], Ralph Lauren debuts 'window shopping' touch screen, http://www.usatoday.com/tech/news/techinnovations/2007-06-20-ralph-lauren-window-shopping_N.htm.

TRUONG, K., E. HUANG, AND G. ABOWD [2004], CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home, *Proceedings of the 6th IEEE Conference on Ubiquitous Computing*, 143–160.

TSICHRITZIS, D., AND A. KLUG [1978], The ANSI/X3/SPARC Framework Report of the Study Group on Database Managmeent Systems, *Information Systems* **2**, 173–191.

TURNER, V. [1975], *Dramas, Fields, and Metaphors: Symbolic Action in Human Society*, Cornell University Press.

TYCHSEN, A., M. HITCHENS, T. BROLUND, AND M. KAVAKLI [2005], The Game Master, *Proceedings of the Second Australasian Conference on Interactive Entertainment*, Sydney, Australia, Creativity & Cognition Studios Press, 215–222.

UBIQ'WINDOW [2008], Ubiq'window Optical Touch Screen Technology.

UBISENSE [2008], Ubisense Real-time UWB Lcation Technology, http://www.ubisense.net.

VIDEOMINING [2008], Video Mining, http://www.videomining.com.

VOCATION VACATIONS [2008], Test-drive Your Dream Job, http://vocationvacations.com/.

VOGEL, D., AND R. BALAKRISHNAN [2004], Interactive Public Ambient Displays: Transitioning from Implicit to Explicit, Public to Personal, Interaction with Multiple Users, *Proceedings of the 17th ACM Symposium on User Interface Software and Technology*, Santa Fe, USA, 137–146.

W3C [2008], Date and Time Formats, http://www.w3.org/TR/NOTE-datetime.

WAGNER, D., T. PINTARIC, F. LEDERMANN, AND D. SCHMALSTIEG [2005], Towards Massively Multi-user Augmented Reality on Handheld Devices, *Proceedings of the Third International Conference on Pervasive Computing*, Lecture Notes in Computer Science, Springer, 208–219.

WAKKARY, R., AND M. HATALA [2006], Ec(h)o: Situated Play in a Tangible and Audio Museum Guide, *Proceedings of the 6th ACM Conference on Designing Interactive systems*, University Park, USA, 281–290.

WAKKARY, R., M. HATALA, R. LOVELL, AND M. DROUMEVA [2005], An Ambient Intelligence Platform for Physical Play, *Proceedings of the 13th ACM International Conference on Multimedia*, Singapore, 764–773.

WEAL, M., AND E. HORNECKER [2006], Requirements for in-situ authoring of location based experiences, *Proceedings of the 8th ACM Conference on Human-Computer Interaction with Mobile Devices and Services*, New York, NY, USA, 121–128.

WEAL, M., D. MICHAELIDES, M. THOMPSON, AND D. DEROURE [2003], The Ambient Wood Journals: Replaying the Experience, *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, New York, NY, USA, 20–27.

WEISER, M. [1991], The Computer of the 21st Century, *Scientific American* **265**, 94–104.

XPATH [2008], XML Path Language (XPath), http://www.w3.org/TR/xpath.

XQUERY [2008], XQuery Language, http://www.w3.org/TR/xquery/.

XQUERYUF [2008], XQuery Update Facility, http://www.w3.org/TR/xquery-update-10/.

XUPDATE [2008], XUpdate XML Update Language, http://xmldb-org.sourceforge.net/xupdate/.

YAHOO [2008], Yahoo Pipes, website, http://pipes.yahoo.com/pipes/.

ZUBOFF, S., AND J. MAXIM [2002], *The Supporty Economy*, Viking Books.

End-user Programming of Ambient Narratives for Smart Retail Environments

Ambient Intelligence is a vision on the future of the consumer electronics, telecommunications and computer industry that refers to electronic environments that respond to the presence and activity of people and objects. The goal of these intelligent environments is to support the performance of our everyday activities using technology and media in way that puts users in control. Many research prototypes and demonstrations of ambient intelligence systems and applications have been developed since the introduction of this vision, but many of these examples focus on a relatively small application domain and set of functionality. The downside of this reductionist approach is that it surpasses the open-ended dynamic nature and complexity that is inherent to social environments.

This thesis aims to find a generic interaction concept to capture the way we form experiences in our everyday life and integrates technology and media into that process. It proposes the design of an end-user programming environment that supports retail designers without special programming skills to create, simulate and deploy smart retail environments within this interaction concept.

To derive such a generic interaction concept it is necessary to look at the social, cultural and economical factors that shape the ambient intelligence landscape to better understand how ambient intelligence helps people in performing their everyday life activities and rituals. This analysis shows that notions like play and performance are not just seen on stage in the theatre: Everywhere around us people perform culturally defined social scripts, e.g. in court, in a restaurant or on the street. Social interaction itself in fact can be seen as an improvised performance that takes shape through the interaction of people with their surroundings. This implies technology can be applied to detect these social scripts and in turn affect the environment to improve the performance. This can be found back in shopping malls and shops for example: The shop employees perform a service for the customer in which the shop itself is seen as the stage and the increasingly interactive, dynamic lighting, audio, video the special effects to enhance the shopping experience for customers. In this experience economy, next to the multi-sensory trend also a development towards co-creation environments can be seen in which the consumer becomes an active producer in the construction of the experience that is offered to him. By looking at such co-creation environment from a literary, semiotic perspective they can be considered as interactive narratives consisting of interrelated social scripts with associated device actions. Through interaction with this possibly dynamically changing *ambient narrative*, i.e. by performing social scripts, players or readers construct their own story or ambient intelligence experience.

To apply this ambient narrative concept in practice in the domain of retail, user research was conducted through interviews and workshops with professional users, i.e. retailers, designers and consultants to discover the types of interactive multi-sensory retail experiences these people would want to see and the requirements placed by this group on an end-user programmable ambient narrative system. From this research we learned that designers had a preference for a 3D simulation on a PC in their office to design ambient narratives, while retailers preferred a PDA version to adjust the retail experience on location. Furthermore, a list of thirty requirements was derived that can be grouped into four categories: *ambient narrative concept implementation*, *run-time system performance*, *functionality end-user programming environment* and *system extensibility*.

On the basis of these findings, a formal model was defined to describe the problem of reconstruction ambient intelligence from the (dynamically changing) modular fragments of which a (dynamic) ambient narrative consists. Each fragment consists of a description of the social script and the actions on devices that are coupled to this script. The author of the ambient narrative can specify which fragments may be activated when by setting event triggers in the action section of another fragment. This model can be represented in a hypertext model in which every fragment is a node and each event trigger a link that connects nodes. An *ambient narrative engine* continuously sequences these fragments based on contextual information, session state and user feedback into a coherent story that is rendered by the devices that surround people. Next to the design and implementation of the ambient narrative engine, the functional user requirements were used to compose a system architecture for an *intelligent shop window* that supports the entire lifecycle of ambient narratives, from the initial design in a 3D simulation until the modification of fragments in-situ with the PDA.

The intelligent shop window prototype with authoring environment was realized and evaluated in ShopLab at the High Tech Campus in Eindhoven on three main criteria: *usability*, *run-time performance* and *extensibility*. To test

the usability a user study was conducted in which the participants were asked to perform four different programming tasks with this prototype tool and fill in a questionnaire afterwards with both questions on the intuitiveness, expressive power and efficiency of both authoring tools. From the results of this study we conclude the target user group is able to design intelligent shop window ambient narratives with this prototype system. An observation-in-use experiment revealed the response time of the system was satisfactory in the majority of cases but in some special cases could lead to long response times as a result of the current implementation of the ambient narrative engine. In terms of system architecture and extensibility towards other applications and domains, the strongest restrictions were placed by the assumed fixed sensor infrastructure and partial implementation of the formal ambient narrative model in the prototype.

Curriculum Vitae

Mark van Doorn was born in Veghel, the Netherlands, on November 23, 1975. After graduating from Gymnasium Bernode, Heeswijk-Dinther in 1994 he moved to the University of Twente, Enschede to study Computer Science. In the winter of 1997/1998 he did a five-month internship at British Telecom Laboratories, Ipswich, United Kingdom. He wrote his Masters Thesis "Thesauri and the Mirror Retrieval Model" and graduated in 1999 at the University of Twente. In 1999 he became a permanent staff member of the User Experiences department at Philips Research Laboratories Eindhoven, where he did his PhD research. Currently he is a research scientist and project leader working on authoring immersive interactive environments. His research interests include end-user programming, interactive storytelling, mixed reality and user-driven design. He has filed over 20 patent applications and his work has been published and presented at several international conferences.