JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, September 2004

Representation Language for Evaluating Reusable Launch Vehicle Concepts

Jeremy Vander Kam *
NASA Ames Research Center, Moffett Field, California 94035-1000

and
Peter Gage†
ELORET, Sunnyvale, California 94087-4202

Evaluation of launch vehicle concepts requires analysis by experts in vehicle sizing, trajectory, propulsion, subsystems, operations, cost, and reliability. A common data model is required to ensure consistent analysis across the various disciplines. This paper describes the development of the Launch Vehicle Language to support evaluation of Second Generation launch vehicle concepts. Vocabulary definition, development of data structures, and implementation in XML are discussed. Future enhancements to facilitate interdisciplinary data exchange are described.

I. Introduction

LAUNCH vehicle architecture evaluation is inherently multidisciplinary. Top-level requirements for NASA's Next Generation Launch Technology program (NGLT)[1] address safety and reliability, cost and mission performance. Discipline experts in Vehicle Sizing, Trajectory, Propulsion, Operations, Cost and Reliability analyze different aspects of the architecture but they are not independent. The Advanced Engineering Environment (AEE) is being developed as an evaluation framework under the Integrated Systems Engineering and Analysis group.[2] The AEE framework must manage interdisciplinary data transfer effectively.

Software systems for multidisciplinary aerospace design have long used a common data model to ensure consistent representation of a particular concept across different disciplinary analyses and to permit consistent comparisons across different concepts.[3-6]

The data model must incorporate a vocabulary that captures all features that are shared between disciplines. The vocabulary must be unified so that terminological differences between disciplines are reconciled and each feature is represented only once. Once a common vocabulary is constructed, each disciplinary analysis need define only a single interface instead of customized interfaces for each other analysis. For N tools, the number of interfaces to be maintained is reduced from $O(N^2/2)$ to $O(N)$ as shown in Figure 1.

While a simple list of data items can support analysis interfacing, the addition of structure can increase the utility of a data model. Organization makes the model more flexible and extensible.[5] Capturing relationships between items facilitates navigation and search and can increase understanding of interdisciplinary influences.[6] Data formatting can improve comprehension for human analysts and facilitate presentation of multiple views of the system being described. An object-oriented model increases the potential for reuse of data chunks from previous designs when preparing to analyze new concepts. Inheritance and encapsulation gained from an object-oriented approach lessen the time spent compiling an assessment using this data model.

This paper reports on the development of the Launch Vehicle Language (LVL) for Systems Analysis of Reusable Launch Vehicles. The data requirements for the evaluation environment are briefly described in the next section. Subsequent sections discuss the development of data content, data structure and operational implementation.

Presented as Paper 2003-1330 at the 41st Aerospace Sciences Meeting, Reno, Nevada, 6-9 January 2003; received 6 November 2003; revision received 7 June 2004; accepted for publication 13 July 2004.
*Aerospace Engineer, NASA Ames APS Branch, MS 258-1. AIAA Member.
†Technical Director, 690 West Fremont Avenue. AIAA Associate Fellow.

The paper concludes with a discussion of plans for further developments that will facilitate continuing evolution of the data model.

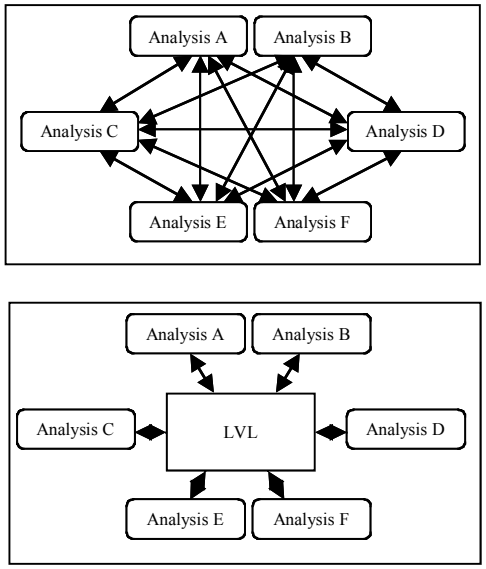


Fig. 1 Benefit of LVL to tool interfacing. Number of interfaces reduced to O(N).

II. Data Requirements for the Advanced Engineering Environment

The Advanced Engineering Environment development plan includes the goal:

To ensure the computing environment readily enables and accommodates all of the engineering/physics-based, cost/economics analyses, reliability/maintainability/supportability (RMS), and operations (flight and ground) tools/methods necessary to fully assess/verify the designs at the architecture, segment, element subsystem, assembly and component levels.

Two major issues arise when developing such a data model. The first is the sheer number of uniquely described inputs and outputs that exist for each analysis tool involved in the assessment. The second is the inherit differences in the data that describe each of the wide range of launch architecture concepts. In order to address these issues, the AEE data model must capture the vocabulary, or data items, from all of the engineering, cost and economics, RMS and operations methods used in design assessment. Construction of the vocabulary is discussed in Sec. III.

The evaluation environment must support analysis of many launch vehicle architectures, each performing several different Design Reference Missions.[1] Each architecture may use a variety of boosters and payload containers, with different elements being used for different missions as shown in Fig. 2. The data should be structured to facilitate the representation of many missions and the associated flight hardware. Furthermore, the environment must support assessment at the "segment, element subsystem, assembly and component levels", so the data structure should reflect that hierarchy. The LVL structure is discussed in Sec. IV. While tailored for the AEE, its basic structure is suitable for general multi-disciplinary analysis activity.

In any multi-disciplinary activity data are operated on and manipulated by multiple parties. Many disciplines interact on the common data set in serial, parallel and iterative fashion. Evaluation set-up, tool interfacing, data reconciliation and data reporting are operations that arise in such activities. Particular assumptions and ground rules must be established to manage the integrity of the common data.

The primary AEE analysis process[2] is presented in Fig. 3. The Setup activity produces an iLVL document, or instance of the Launch Vehicle Language. Tool interfacing is managed through iLVL documents. Parallel execution of Operations, Cost and Safety produces three iLVL documents that must be reconciled into a single unified document. After analyses are completed, contents of the final iLVL document are manipulated for data summary and review.

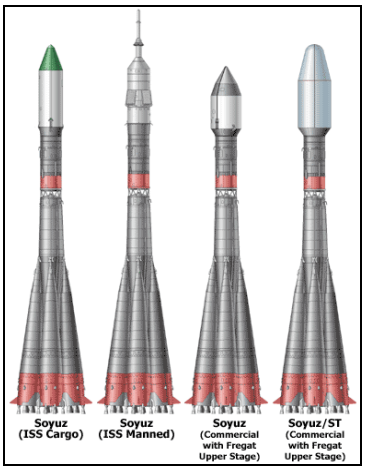


Fig. 2 Launch architecture modularity.

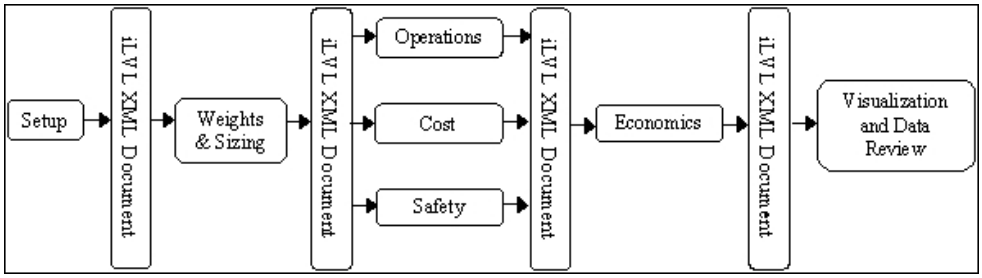


Fig. 3 Primary AEE process flow.

A modular data structure facilitates Setup, because large chunks of data from previous assessments can be assembled into a complete data set for a particular analysis. Similarly, at the termination of an evaluation the chunks may be saved into the library for later use. These pieces may reflect the physical hierarchy of the launch system, as discussed earlier, but we may also develop chunks describing missions, technologies and operating assumptions. During the formulation phase of an analysis process, this library may be accessed by anyone on the analysis team to assemble the vehicle, mission, operations approach, ground rules and goals of interest.

Tool interfacing can be automated when the common data items are exactly specified. Commercial software used in the Advanced Engineering Environment supports automatic link generation when data names are identical.[11] This specification is somewhat complicated when the data model is object-oriented because the detailed structure is different for different analysis cases and the data names are therefore context-dependent. Tool interfacing is discussed in the Implementation section of this paper and ongoing work in automated linking is described in the Future Work section.

Data reconciliation is facilitated when data items are tagged with "ownership" information. Discipline tools may only write to the areas of the data set that fall in their sphere of responsibility. The reconciliation method described in the Current Implementation section relies on data ownership being partitioned between analyses that execute in parallel.

Reporting of results requires reformatting of the data generated during analysis. The Advanced Engineering Environment supports a number of pre-specified report templates, which have formats ranging from web-served HTML to Excel spreadsheets. The utilities currently used for report generation are described in the Current Implementation section.

III. Vocabulary Development

Across the aerospace engineering community, launch vehicle architecture data are defined in different ways. Sometimes different groups assign different names to the same item: Reference Area or Planform Area or Sref may all refer to the area of the wing projected onto a horizontal plane. In other cases, different groups may use the same

term to refer to different entities: one group may calculate Wing Area including carry through area that is inside the fuselage while another group uses only the area that is exposed outside the fuselage. A common vocabulary reduces the difficulty of maintaining these interfaces between tools. Instead of building a customized interface to communicate with each other tool, a discipline can use a single interface through which it can communicate with all other tools.

A simple list of data names is not sufficient to construct a common vocabulary, because some explanation of the data items is needed to reconcile ambiguities. The descriptors listed in Table 1 were solicited from discipline experts for each data item. Units and Type help in tool interfacing. Data Supplier and Data Customer support reconciliation and reporting functions.

Table 1 Data descriptors in the LVL

Descriptor	Type	Example
Item Name	string	ExposedWettedWingArea
Description (text)	string	The wetted area of the wing that is not contained by the fuselage.
Units	string	$meter^2$ or ft^2
Type	string	double
Data Supplier	string	Geometry
Data Customer	string	Weights, Sizing, Cost
Information Source	string	Part Drawing #2112

The LVL supports three basic types of items: Variables, Arrays and Groups. The descriptors in Table 1 apply to all Variables and Arrays. Groups are containers that hold Variables, Arrays or additional Groups. These basic building blocks are arranged into many different sub-structures in the LVL. These include arrangements that represent Components, Trajectories and Aggregations, each of which is discussed in the following sections.

Several methodologies are available for working with a team to construct a data model. Initially we adopted a Facilitated Team approach by asking discipline experts from NASA's Inter-Center Systems Analysis Team (ISAT), who would be end users of the data model, to list all data items that they received from an external supplier or sent to an external customer. A known disadvantage of this approach is that it may "require a greater commitment of organizational resources than…justified by the benefits or return on investment for the project".[7] The heavily loaded ISAT team could not devote the necessary resources to make a comprehensive response, so in many cases we took input and output data listings for existing software and filled in the attributes as best we could. This methodology was closer to the Chief Engineer approach, where the designer gathers requirements and data samples and constructs a data model from them.[7] Cognizant of the dangers of excluding the end users from the development activity, detailed feedback and suggestions from end users is collected and implemented in frequent updates to the model.

Once the raw data has been collected it must be reconciled into a consistent set. This is a matter of painstakingly comparing the data lists from each discipline. Iteration with the discipline experts is required to resolve ambiguities. As an illustration of the complexity of this process, consider the analysis tools used by the set of ISAT disciplinary analysts. The analysts in this set make use of 14 discipline models, which incorporate 22 separate analysis tools that make 6340 links involving 13192 variables. The discrepancy between the number of links and variables results from the fact that some data items of one tool may only be obtainable by applying a function of several others. For a new group of analysts the number of these variables that are similarly named or defined is a small fraction. Simple translations such as matching Sref with PlanformArea as well as complex differences in vehicle WBS definitions must be reconciled. The initial formulation of the data model is a tedious task of eliminating redundancies and specifying comprehensive definitions. The result of this process for the AEE is the Launch Vehicle Language that specifies each variable using the metadata in Table 1.

IV. Data Structure

Once the vocabulary is defined, a structure may be implemented to capture relationships between data items. Structure in the LVL enables several policies and capabilities that are unavailable in a "flat" scheme. A well-defined structure such as that of the LVL enables elements to appear in multiple locations of the data model and still be uniquely defined. The tree structure corresponds to the hierarchical design approach that is used in the launch vehicle application, and "almost universally" in design.[9] The relative location of data items conveys important relationships between them.

This structure is the product of several revisions. Important functional relationships are captured. Fig. 4 illustrates a basic separation that is made between the intrinsic properties, or form, of the hardware (DryElements) and the instance-dependent properties associated with the functional operation of that hardware (Mission). Data substructures for both DryElement and Mission are listed in Table 2.

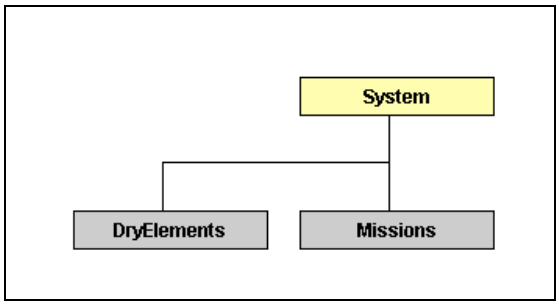


Fig. 4 LVL Top-level structure partitions the constant DryElement properties from the mission-dependent properties.

Table 2 Scheduled LVL substructures

Sub-Structure	Definition	Example
DryElement	The Physical parts of a vehicle stage that do not depend on the mission being flown. Contains a tree of Components.	Shuttle Orbiter
Component	A Physical part of a Dry- or FlightElement.	Fuel Tank
FlightElement	The physical parts of a vehicle stage that are mission-dependent	Shuttle Orbiter + Propellants, Expendables, Crew, Payload
Mission	Definition of the mission to be assessed	ISS Re-supply from KSC
ElementSet	Any combination of one or more FlightElements	ShuttleOrbiter + ET + 2 SRBs
Trajectory	A continuous path that any FlightElement can be tracked through during a mission. Consists of Segments and Events.	Path of Shuttle Orbiter from Launch to MECO
Segment	A portion of a Trajectory flown by an ElementSet. Bounded by Events.	Path from Launch to staging of the SRBs
Event	Bounds Segments. Denotes some change in vehicle configuration, evaluation technique, etc.	Staging of the SRBs

Each mission may use different combinations of vehicle stages. Figure 5 provides an example of an architecture with 4 different DryElements and two missions that use different subsets of those elements. A common booster and orbiter that might carry either a payload pod or a CTV to orbit would be represented in this manner.

Tracking the complete hardware layout for the same stage multiple times in a single iLVL causes confusion and considerable redundancy. To combat this, vehicle stages are separated into DryElements and FlightElements. DryElements contain the hardware breakdown that does not depend on the mission being flown while FlightElements contain the components such as payload, crew and fluids that do depend on the mission context. Each Mission then includes a set of FlightElements that references some number of the DryElements in the architecture. These FlightElements can then be assigned to various portions of the Mission Trajectory structure as illustrated in Fig. 6.

The primary goals of the LVL structure are to maximize re-use and minimize redundancy. Divisions are made based on the typical needs and understandings of analysts in the field. The current LVL is divided according to the physical and performance and life-cycle aspects of the launch system. Elements correspond to the form of the system, Missions describe function and the Cost, Operations, and Safety Reliability groups characterize behavior.

Such an arrangement is consistent with NIST Design Repository activity, which uses an "object-oriented artifact representation language that provides a high level division into form, function, and behavior."[6]

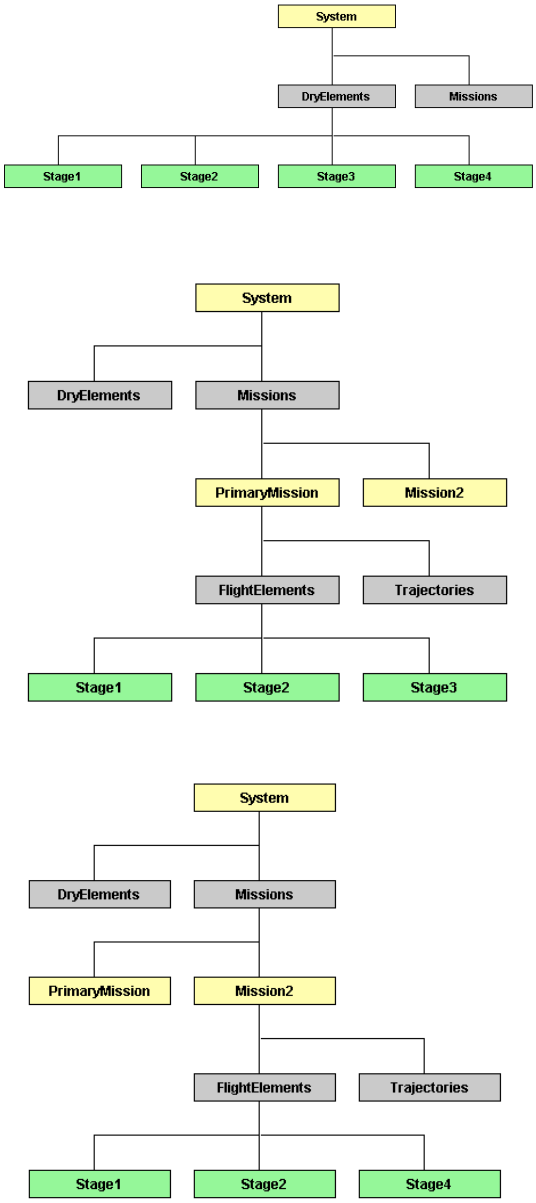


Fig. 5 DryElements and FlightElements.

Each DryElement and FlightElement contains a tree of recursively defined Components that represent the physical pieces that make up the Element. Each Component carries a Weight that is defined in one of two ways. The Weight is either the sum of the child components or (if the Component has no children) the resolution of a weight equation or WER (Weight Estimation Relationship). The general Component structure and its possible content is presented in Fig. 7. A minimal definition requires (denoted by solid boxes) the number of instances of that Component and the Weight of a single instance of that Component. Optional data items, including Reliability, Cost, and Operations data, are denoted with dashed boxes in the figure. Each of these sections has its own modular definition that may be applied to data nodes other than Components as well. The recursive use of Components covers the Element Subsystem, Assembly and Component hierarchy that is required for AEE and generally used by the community at large, but is extensible to an arbitrary number of levels. An important feature is that the LVL does

not mandate any ordering of Components. One is free to construct a WBS with an arbitrary number of Component layers in the hierarchy.

The basic Component specification is extended to represent specialized components with particular data needs. Wing, Engine, and FuelTank are examples of such specialized components that are implemented in the current LVL. These Components require additional detailed data such as geometry to uniquely describe them.

While the basic structure of the LVL model is a hierarchical tree, it is important to incorporate data references to eliminate redundancy even across branches. For example, consider the FlightElement, which contains the physical components of a vehicle stage that depend on the mission being flown. These may be fluids, payloads and crew. Each FlightElement references a DryElement in the DryElements section of the iLVL. When an iLVL contains several Missions, certain FlightElements may appear many times as they are involved in many Missions. The use of references within the iLVL permits a single specification of a DryElement and a limitless number of references to that specification. This use of references prevents the introduction of inconsistent specifications of components that are intended to be identical. It also substantially reduces the size of the iLVL document.

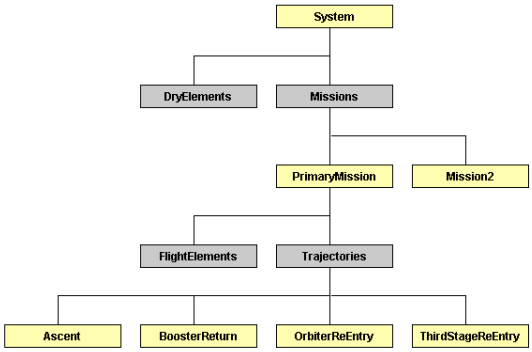


Fig. 6 Missions and trajectories.

Another implementation of references is seen in the Aggregation sub-structure. Aggregations represent quantities derived from other data items in the iLVL. That is, they specify a relationship of values that does not follow the hierarchal tree structure. They consist of an equation and a group of coefficients that are involved in the equation. Each coefficient is either assigned a value, or is specified by reference to another value in the iLVL.

Consider the following example:

The total engine DDT&E cost of the ascent elements of an architecture is the sum of the engine DDT&E cost of the booster and the orbiter. These two parameters exist in different branches of the LVL structure – in the DryElement/Cost groups of their respective stages. The aggregation that defines the ascent element DDT&E cost would be defined in an iLVL as

```
<Aggregation name=”EngineDDT&ECost”>
<Coefficients>
<Variable name=”boostEngCost”> System/DryElement[@name=’Stage1’]/Engine/DDTE
</Variable>
<Variable name=”orbEngCost”> System/DryElement[@name=’Stage2’]/Engine/DDTE
</Variable>
</Coefficients>
<Equation>boostCost + orbCost</Equation>
</Aggregation>
```

This excerpt from an iLVL corresponds to the concept expressed in Fig. 8.

The Aggregation construct is also used to define the Component WERs where applicable. In the WER example, the Component weight is found through the resolution of an equation that may reference geometric properties of the Component. For example, wing weight may be a function of the wing aspect ratio, wetted area etc. An equation is constructed for the weight of the wing that consists of coefficients for these parameters. These coefficients in the iLVL document contain XPath pointers to the values of the wing aspect ratio, wetted area and so forth. This allows parametric changes to propagate throughout the document consistently. Redundancy is eliminated as each value appears only once in an iLVL no matter how many equations utilize it. This reduction in redundancy is available,

but not enforced in the LVL. Real-world applications are permitted to set up an iLVL with as few or as many redundant definitions as desired according to the needs of the application. To avoid unintentional redundancies, users can search for existing instances of a coefficient in an iLVL. When a successful match is identified, the user can specify a link to the existing item rather than supplying a new value.

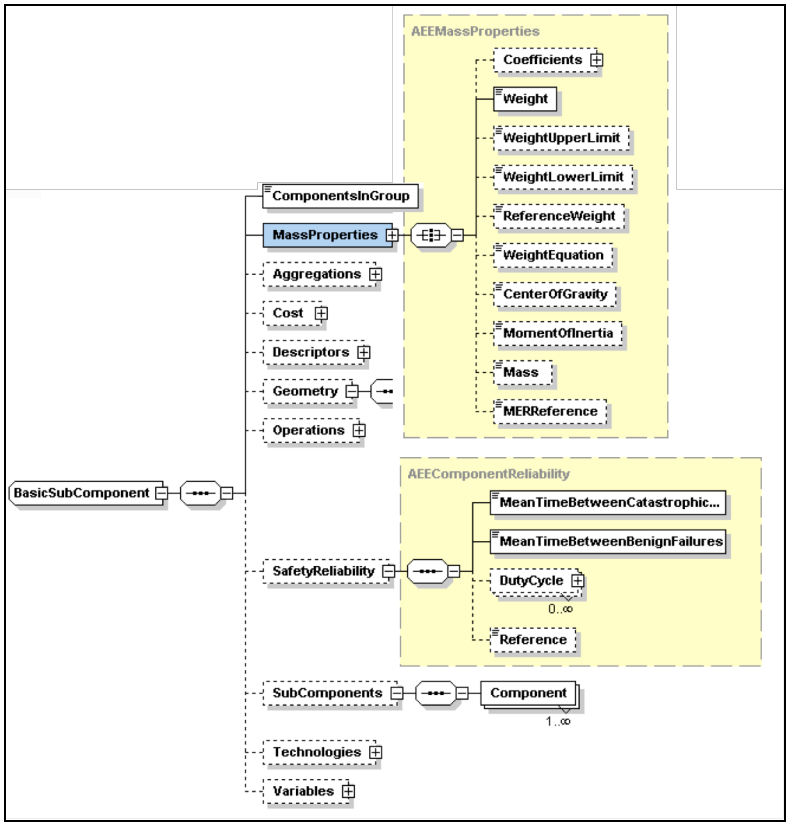


Fig. 7 Basic component definition.

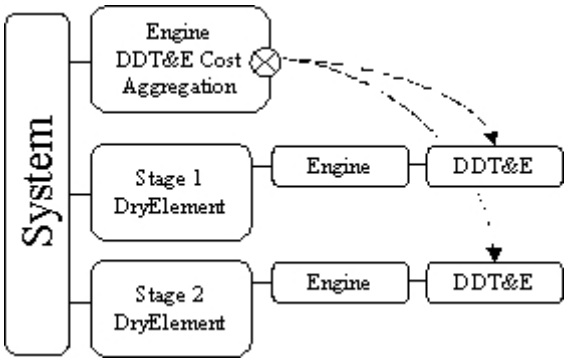


Fig. 8 Aggregation functionality.

V. Current Implementation

A. Data Model Specification

LVL is implemented as a domain-specific extension to XML. Initial attempts to assemble a common vocabulary for AEE users were conducted using Excel spreadsheets that contained the descriptors listed in Table 1. As responses were received from different participants, we found that the set of descriptors needed to evolve, and it was

difficult to manage the changes in Excel. Custom scripts were developed to merge data from the independent responses. A more robust data representation scheme was required. Table 3 lists the features that drove selection of XML for LVL implementation.

XML is very commonly used as an interchange format for linking unlike systems.[8,9] The basic tree structure of an XML document corresponds to the data hierarchy that we have developed. XPath allows references to be specified within a document. Open source tools such as the Apache Xerces and Xalan[10] Java packages are used for manipulation, transformation and service of the iLVL documents. XSL style sheets provide further transformation and report generation.

LVL has already improved data integrity and consistency for launch vehicle analyses. iLVL documents are validated against the LVL specification using XML Schema. Each data item in an iLVL is stored once and referenced, so data editing is efficient and reliable.

Table 3 Important features influencing choice of XML for LVL implementation

	Implementation Alternative		
Feature	**Plain Text**	**Excel**	**XML**
Open standard format	No	Proprietary standard	W3C specification
Structure, hierarchy	No	Limited	Yes
Extensible	Yes	Yes	Yes
Validity checking	No	Limited	Yes
Style transformation	No	Limited	XSL
Link within document	No	Limited	XPath
Development tools available	No	No	Xalan and Xerces
Report generation	No	Yes	Yes

B. Setup

At the beginning of the assessment activity, the iLVL is constructed and "primed" using a Setup application. This application enables the user to construct the iLVL from a library of existing data chunks and to assign values for individual data items in bulk based on choices about the architecture type, mission objectives, included technologies and tool suite. This library of items contains 'valued' data so that selecting a particular item, say a type of Mission, imports not only the appropriate structure, but the relevant values for entire sub-trees. Figure 9 illustrates the use of the LVL Setup application. In the example of Fig. 9, the user selects the Stage 1 DryElement and chooses to add a Sibling Element (an Element at the same level of the hierarchy). The application pops up a window that lists all DryElements available in the library. The user selects NASARefNCFOrbiter, which is then inserted into the architecture tree. If the user wants a DryElement that is the same as NASARefNCFOrbiter except that it uses different engines, the Engine Component present in the DryElement is deleted and replaced by another selection from the library.

The LVL Setup application also provides an interface for editing individual data items. Instead of deleting and replacing the Engine Component, the user could navigate to the individual items for which values need to be changed, and edit them through a Popup window. WERs and Aggregations can also be edited. A user may express an equation and then click on the appropriate coefficients of the equation to define the items they describe.

C. Interfacing

Once the iLVL is constructed, it is provided to the various analysis tools using the Phoenix Integration Model Center application.[11] In this application the analysis tools are wrapped so that their inputs and outputs are exposed to the ModelCenter environment. Several applications provide exposure and interfacing between the iLVL and the tools in a given assessment. A global data model enables an application to use a set of rules about a specific analysis tool to discover the contents of any iLVL document that are pertinent to that tool and link them together. In the current implementation, each disciplinary analysis tool has an interface defined to the areas of the LVL that it consumes and provides data to and from. This interface is controlled through the link set of each Model Center model. When each analysis completes, its results are published to the AEE PDM for use by other tool sin the assessment. The PDM controls the order of execution of each analysis.

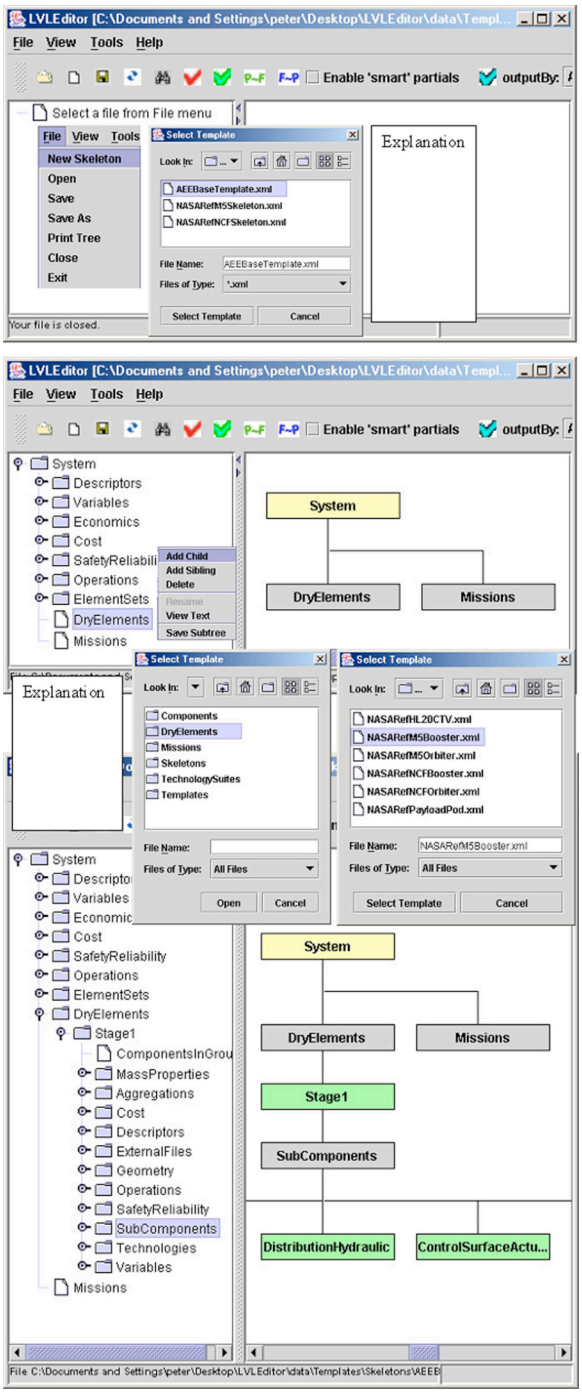


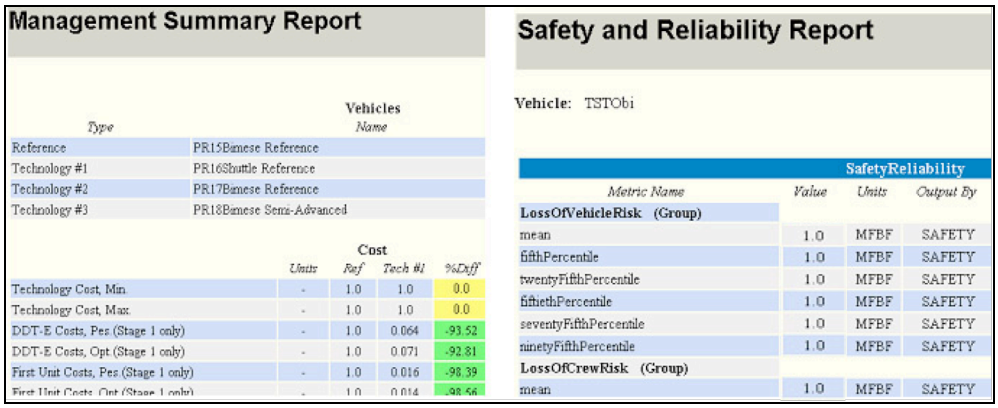
Fig. 9 iLVL Construction in the LVLEditor.

D. Reconciliation

As seen in Fig. 3, some disciplines conduct their evaluations in parallel. When this occurs, their results must be reconciled into a single result. This is automatically accomplished using applications that take advantage of the Data Supplier (outputBy) attributes of each variable and array. Each of the parallel tools populates a copy of the process iLVL with its outputs, each of which are combined into a single result. Each discipline tool may only have write access to the portions of the iLVL that fall under that tool's area of relevancy.

E. Reporting

At the end of the process, several reports are available. These reports can be defined by specific values or focus on items within certain divisions of the LVL. Cost, Operations and Economics reports are available as well as summary reports that bring data from all of these disciplines together as shown in Fig. 10. The content generated by any particular assessment is archived on the AEE PDM as iLVL documents. Each report format is driven by the content of these iLVLs.



Management Summary Report

Vehicles

Type	Name
Reference	PR15Bimese Reference
Technology #1	PR16Shuttle Reference
Technology #2	PR17Bimese Reference
Technology #3	PR18Bimese Semi-Advanced

Cost

	Units	Ref	Tech #1	%Diff
Technology Cost, Min.	-	1.0	1.0	0.0
Technology Cost, Max.	-	1.0	1.0	0.0
DDT-E Costs, Pes (Stage 1 only)	-	1.0	0.064	-93.52
DDT-E Costs, Opt (Stage 1 only)	-	1.0	0.071	-92.81
First Unit Costs, Pes (Stage 1 only)	-	1.0	0.016	-98.39
First Unit Costs, Opt (Stage 1 only)	-	1.0	0.014	-98.56

Safety and Reliability Report

Vehicle: TSTObi

	SafetyReliability		
Metric Name	Value	Units	Output By
LossOfVehicleRisk (Group)			
mean	1.0	MFBF	SAFETY
fifthPercentile	1.0	MFBF	SAFETY
twentyFifthPercentile	1.0	MFBF	SAFETY
fiftiethPercentile	1.0	MFBF	SAFETY
seventyFifthPercentile	1.0	MFBF	SAFETY
ninetyFifthPercentile	1.0	MFBF	SAFETY
LossOfCrewRisk (Group)			
mean	1.0	MFBF	SAFETY

Fig. 10 Sample LVL reports.

VI. Future Directions

A. Evolution

LVL evolution continues to support emerging and legacy tools. Data covering all aspects of the vehicle cycle from wheel stop to launch are being added. Extensions for non-physical processes and requirements are under consideration. This includes ground operations and system requirements objects such as figures of merit. Incorporation of aircraft-related data is ongoing. All of these additions enable a greater user base. To date, these additions have required little fundamental change to the LVL hierarchies – validating that the LVL structures are "good" ones.

B. Setup

The power of the common language is multiplied when many organizations can access a central repository of LVL data. A web-based library of LVL parts and fragments is in development. Any authorized party may then incorporate results expressed in the LVL and published to the library into its own evaluations. This will enable consistent cross-agency data transfer at the analysis tool level. Outside corporations may submit concepts for evaluation by NASA analysts without the need for translation of data formats.

C. Data Binding

As the assessment data grows and general search is required, the iLVL contents can be parsed into a commercial database. Many products are available for XML data binding.[12] Implementation is anticipated in the coming year. Coupling this to the web-based portal of the previous paragraph yields a powerful, query-able common data repository.

D. Interfacing

A general linking approach will enable tool interfaces to take full advantage of the LVL scalability to the limit of the tool itself.[13] Link maintenance will be dramatically decreased by the introduction of a generalized solution. Links will be expressed in general LVL terms, eliminating the dependence on particular identifiers used for particular vehicles and architectures. A single, generic link definition will be possible for each tool. This single definition, expressed according to the LVL schema definitions, will be valid for any instantiation of the language.

E. Utilities

Increased experience enables a large range of utility development. iLVL differencing, tool link tracking, system mapping, custom reporting, automated iLVL updating and powerful iLVL development applications have and continue to evolve.

VII. Conclusions

A consistent and intuitive data model in any multidisciplinary assessment activity is essential. The Launch Vehicle Language (LVL) provides such a data model. Supporting utilities to aid in assessment setup, tool interfacing, data reconciliation and results reporting are in use and are being extended to maximize automation, consistency and integrity of launch vehicle analysis data. The LVL has been adopted by the NASA SLI AEE program and is in use today. Extension to the aerospace community at large will enable data transfer across agencies and corporations in an automated, consistent and intuitive manner. The LVL schemas will be made available publicly via the DoD Defense Information Systems Agency (DISA).[14]

Acknowledgments

The authors were supported through contract NAS2-00062 from NASA ARC Systems Analysis branch. Several developers contributed to the tools that support the LVL in the Advanced Engineering Environment. Min Qu, of AMA Inc, developed the "Data Dictionary Plugin". Leo Fabisinski, of ISSI, developed the autolinker utility. Eric Lau, Xun Jiang, Grant Palmer and Jeremy Wong, of ELORET, contributed to the graphical editor for LVL.

James Reuther, the AEE Technical lead, Donovan Mathias, the Tool Development lead, and the entire AEE team contributed data requirements to the vocabulary and provided feedback on proposed data structures.

References

[1]"The Next Generation Launch Technology Program Goals, Objectives and Figures of Merit," NASA Office of the Chief Engineer, March 2003.

[2]Monell, D., Mathias, D., Reuther, J., and Garn, M., "Multi-Disciplinary Analysis for Future Launch Systems Using NASA's Advanced Engineering Environment (AEE)," AIAA Paper 2003-3428, *16th AIAA Computational Fluid Dynamics Conference*, June 2003.

[3]Rowell, L. F., "The Environment for Application Software Integration and Execution (EASIE) Version 1.0 - Volume 1 - Executive Overview," NASA TM-100573, Aug. 1988.

[4]Cousin, J., and Metcalfe, M., "The BAe (Commercial Aircraft) Ltd Transport Aircraft Synthesis and Optimization Program," AIAA Paper 1990-3295, *AIAA/AHS/ASEE Aircraft Design, Systems and Operations Conference*, Sept. 1990.

[5]Takai, M., "A New Architecture and Expert System for Aircraft Design Synthesis," Stanford Ph.D. Thesis, June 1990.

[6]Szykman, Simon, Bochenek, Christopher, Racz, Janusz, and Sriram, R. D., "Design Repositories: Next-Generation Engineering Design Databases," *IEEE Intelligent Systems and Their Applications*, 2000.

[7]Katz, R. H., *Information Management for Engineering Design*, Springer Verlag, Berlin, 1985.

[8]St. Laurent, S., *XML: A Primer*, 2nd ed., M&D Books, Foster City, CA, 1999.

[9]Navarro, A., White, C., and Burman, L., *Mastering XML*, Sybex, Alameda CA, 2000.

[10]The Apache software group home page, http://xml.apache.org (cited Sept. 2004).

[11]Malone, B., and Papay, M., "ModelCenter™: An Integration Environment for Simulation-Based Design," *Simulation Interoperability Workshop,* Simulation Interoperability Standards Organization, 1999.

[12]McLaughlin, B., *Java and XML Data Binding,* O'Reilly, Sebastapol, CA, 2002.

[13]Vander Kam, J., Gage, P., and Sohn, J., "The Launch Vehicle Language Data Model Applied to Analysis Tool Connectivity," AIAA Paper 2004-0206, 2004.

[14]DoD Metadata Registry and Clearinghouse, http://diides.ncr.disa.mil/xmlreg/user/index.cfm (cited Sept. 2004).