Using the US Department of Defense Architecture Framework (DoDAF) in Reengineering Product Development Information

Henson Graves

Lockheed Martin Aeronautics Company
Post Office Box 748
Fort Worth, TX 76101
817-777-1856
henson.graves@lmco.com

Chris O'Donnell

BAE Systems
Post Office Box 748
Fort Worth, TX 76101
817-777-5763
chris.e.odonnell.itar@jsfmail2.p.external.lmco.com

Russ Campbell

Lockheed Martin Aeronautics Company
Post Office Box 748
Fort Worth, TX 76101
817-763-3894
russell.c.campbell@lmco.com

ABSTRACT

The Department of Defense Architecture Framework (DoDAF) is an effective tool for re-engineering product development information management. Our premise is that technical performance in product development is highly correlated with information management capability. System engineering principles can be applied to analyse product development information management and DoDAF is a good methodology for this analysis. DoDAF can be used to develop better operational requirements and used for gap analysis between requirements and the collaborative environment that implements the requirements. Developing the DoDAF products provides the information needed to develop realistic operational and systems models of the development enterprise. The models can be used to perform a gap analysis between the operational requirements and the system implementation. The principles and methodology outlined here can be used to develop predictive measures of technical performance.

.

1 Introduction

This paper describes a technical approach to improve product development technical performance based on system engineering principles. The approach begins from the premise that an enterprise can work effectively only when complete, accurate, product information is available. While there may be debate as to how to quantify the relationship between poor technical performance and poor information management the relationship is well established. The approach described in this paper focuses on developing forensic tools that can be used to develop requirements for information technology needs, analyze gaps between requirements and information technology infrastructure, and develop predictive metrics for program technical success.

1.1 Applying System Engineering Principles

Our approach applies system engineering principles to the process of product development and the enterprise systems that implement the development process. The approach makes use of methodology and tools that have been developed and used primarily for products rather than the operations and systems used to produce products. Applying engineering principles to analyze and model a product and to model the process that produces the product have some overlap. One could argue that system engineering for product development should naturally include the broader analysis of processes and operations, as well as analysis of the information technology infrastructure. In any case applying system engineering principles to the product development enterprise is critical to program success. To the credit of the principles and tools discussed here we have found that they are equally applicable for product development analysis and design as they are for product analysis and design.

1.2 Enterprise Modelling

Our approach relies on enterprise modeling of product development operations and comparing the operational model with its physical realization in the tools and information technology infrastructure. The operational modeling provides the material needed for information technology requirements analysis. Analysis of how well the system (physical) implements the operations model makes gaps and disconnects visible. The analysis and work products produced in this enterprise engineering effort are the basis for good predictive metrics for program success.

A major difficulty with developing enterprise models is that accurate information about enterprise and the way that it operates is difficult to obtain. Modeling is useful only when it reflects the enterprise being modeled. Enterprise operations documentation often does not have sufficient detail to provide good information technology requirements. The information technology fielded is then insufficient and this fact is often realized only late in the game after the requirements have been understood. Remedying the requirements problem requires a disciplined way to develop requirements and perform gap analysis between requirements and information technology infrastructure.

We have found that the US Department of Defense Architecture Framework (DoDAF) is a good methodological starting point for enterprise modeling and that enterprise modeling is needed to develop and field effective information technology solutions.

1.3 Translating Enterprise Modeling into Action

Our approach, as noted, relies on product development enterprise modeling. Many enterprise modeling exercises do not translate directly into concrete action that improves the enterprise processes. The enterprise models that we develop represent information that management must have to operate effectively; the models identify information technology requirements which if not met lead to cost and schedule overrun. Enterprise modeling can be taken a step further; it can be used to directly configure a Web-based information system. This method of implementation of information management requirements has the advantage of reducing programming costs for information technology infrastructure development and maintenance.

1.4 Look Ahead

Section 2 gives background on how our system engineering approach to product development information management was developed and how we were led to use DoDAF and other system engineering enablers. Section 3 outlines the approach to analyzing a product development enterprise using DoDAF. Section 4 summarizes how DoDAF products translate into metrics for judging product development enterprise maturity with respect to information technology.

2 Applying System Engineering principles to Product Development Information Management

A primary goal for product information management is to establish accurate product baseline representations that can be shared between team members and to make this representation available through a Web-based access system. Further goals are to:

- Provide timely inputs for modeling, simulation, and analysis, resulting in shorter decision cycles.
- Improve the traceability (i.e., validation) of product representations used in program activities.
- Increase information coherency, resulting in the reduction of potential misunderstandings across the product team.
- Reduce manual data translations yielding lower translation costs and increased productivity.
- Provide a common repository of accessible and relevant information throughout the product life cycle, saving resources and facilitating better-informed decisions/actions.

2.1 Correlation between Information Management Capability and Technical Performance

While it may be difficult to establish an exact quantifiable relationship between program performance and technical information management, cost and schedule overrun and product quality are often traceable directly to information loss and corruption within the product development process. Information loss and corruption result from not understanding the information exchange requirements between development teams with a resulting inability to supply the required information when it is needed, in the form that is needed. The inability to supply information in the form needed when needed and resulting latency and rework in the product development process results less from electronic connectivity, than from the inability to locate the information and have it in a usable form. The information needed may well be electronically reachable if only one knew where it was located. The ability to use information successfully depends on using the right information, i.e., from the authoritative source and knowing limitations on its validity and use. **Figure 1**, adapted from a Gartner Group presentation, illustrates that the loss and corruption of information can be an unfortunate side-effect in the product development process.

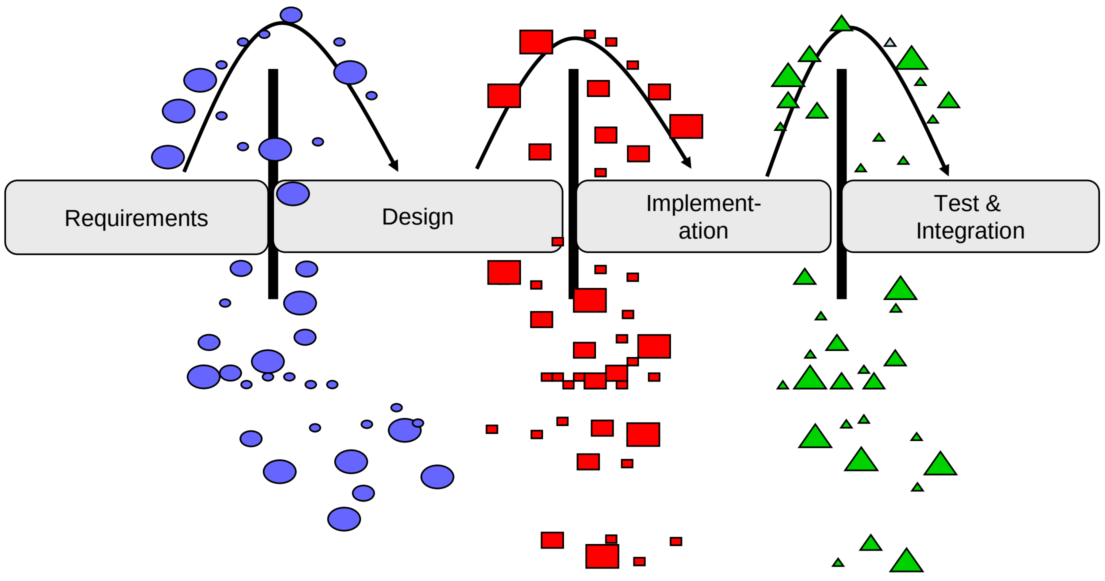


Figure 1: Information Loss and Corruption in Product Development

2.2 Common Causes Affecting Information Management

There are many reasons why it is difficult to establish and maintain product design baselines. Within product development each subject domain such as requirements analysis or design synthesis often uses different tools to produce data and each tool may have its own format. The data produced by these tools overlaps in content, but insufficient checking is performed to ensure consistency. Often insufficient tool-to-tool integration means each user must login to each tool separately and transfer data manually. As a result of insufficient tool integration there is insufficient ability to search across multiple databases. **Figure** 2 illustrates some of the kinds of product data that need to be managed for aircraft development. This data is often stored in separate databases. The lines in **Figure 2** connecting the databases indicate dependences of one kind of data on another kind of data. When changes occur in one database the dependent databases must also be changed to maintain data coherency.

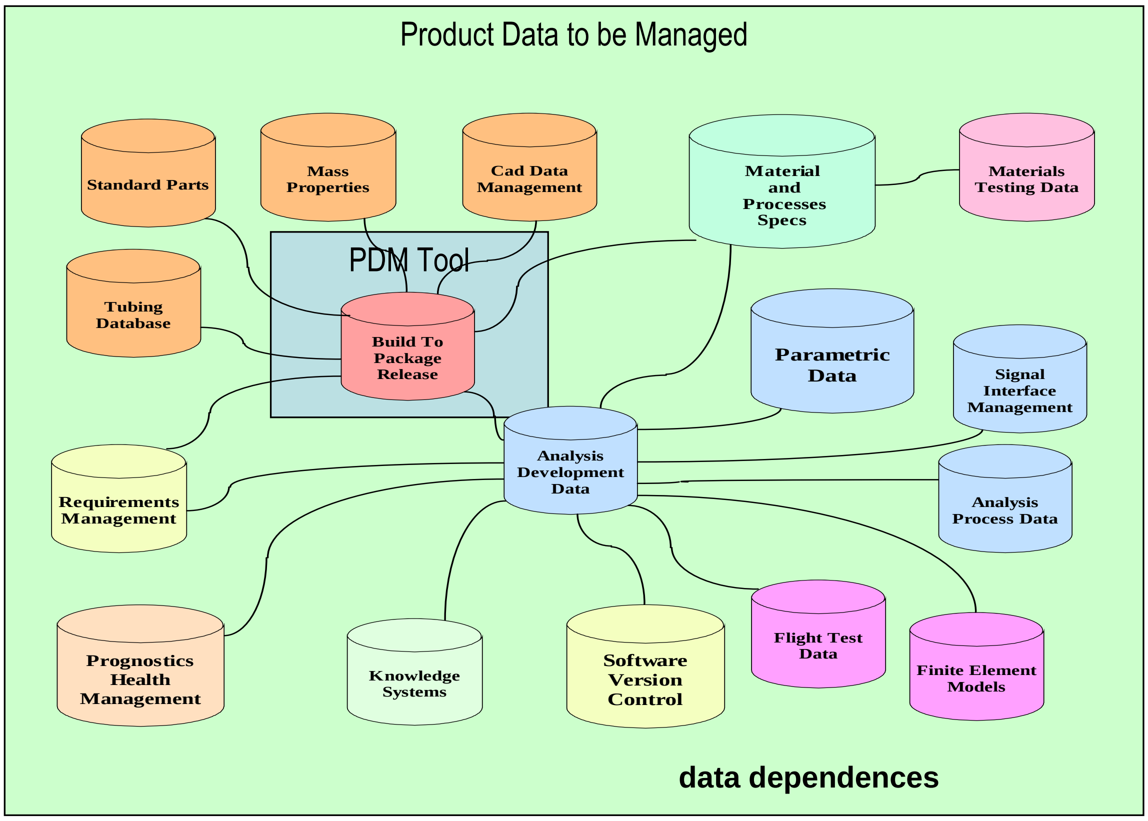


Figure 2: Dependencies within Product Data

2.3 Product Information Management Concepts

Product information management includes many concepts. Some of these concepts are summarized below.

Visibility and awareness: Can users find the information they need and are they aware when the information changes.

Accessibility: Can the information, if found, be accessed.

Understandability: Is the information in an understandable and usable form.

System Interoperability: Can the information be used across different tools and databases.

Integrity: Is the information coherent and can its source can be traced to primary engineering data with any limitations on its valid use identified.

Security and Authentication: Is information access controlled so that only authenticated users that have access permission have access.

Figure 3 illustrates these information concepts.

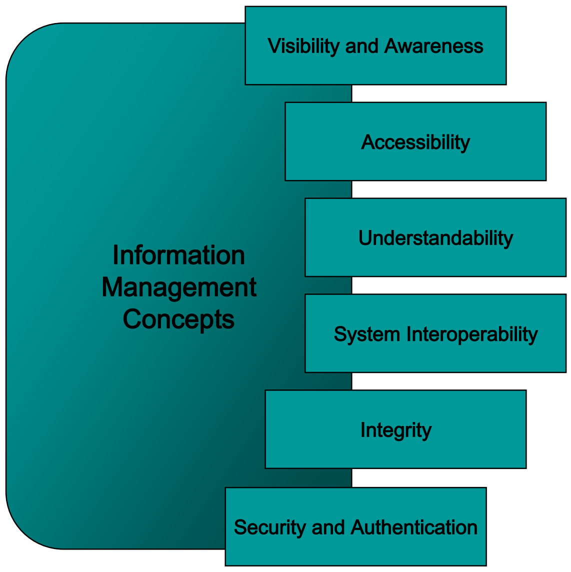


Figure 3: Information Management Concepts

2.4 Technical Approach to providing access to authoritative product data

The goal of an information management system that provides access to authoritative information characterizing a product is easier said than done. First one must define the scope and characterize the data to be managed. The starting point for scoping the data is inclusion of design representations of the product produced by the design teams. Data could be attached to, or referenced by, components of a product decomposition. However, a design effort produces multiple design decompositions, e.g., functional, logical, and physical decompositions, often using different tools that produce data in different formats with different semantics. The information used in product development and evaluation is often system performance data. However, performance data needs to be traceable through intermediate steps to primary engineering and measurement data. While product representations can be configuration managed, understanding the relationships between these different representations is not easily represented within traditional Product Data Management (PDM) Systems as they usually manage data at such a high level of granularity that the detailed relationships are not captured within PDM organizational structure.

2.4.1 Developing Information Management Requirements

Figure 4 illustrates the use of a traditional system engineering approach to analyse requirements for the product development operation, synthesize a system design for product development, and integrated commercial tools to produce the required system.

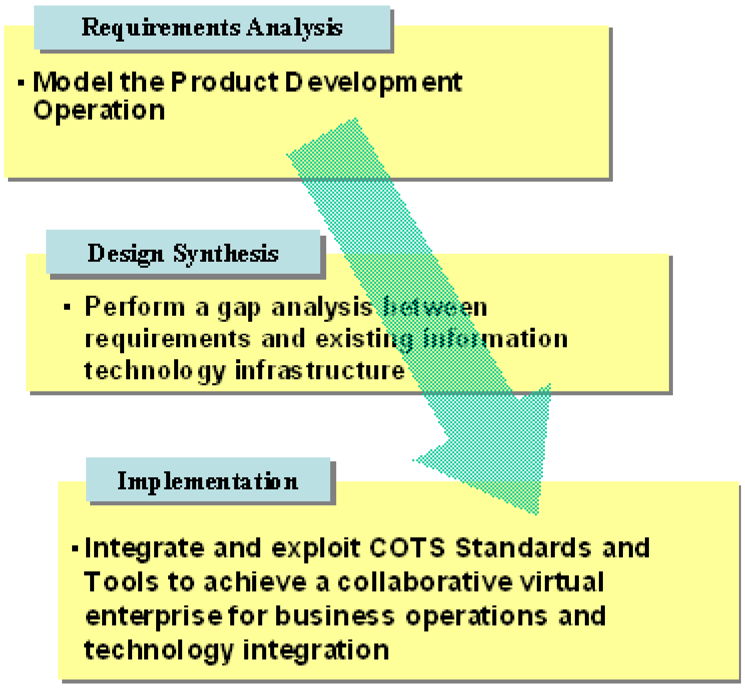

Requirements Analysis
▪ Model the Product Development Operation
Design Synthesis
▪ Perform a gap analysis between requirements and existing information technology infrastructure
Implementation
▪ Integrate and exploit COTS Standards and Tools to achieve a collaborative virtual enterprise for business operations and technology integration

Figure 4: The Engineering Process to Produce an Information Model and a System to Access Product Data

2.4.2 Collaborative Infrastructure Requirements

Many product development efforts have intentionally taken an incremental approach to determining requirements for the collaborative environment, i.e., tools, applications, and network interconnections, and to evolving its implementation. Major programs almost always have legacy information systems; there are prior commitments to specific tools and processes, often without complete understanding of how requirements have changed. Modification to legacy systems often does not come cheap. Any solution to gaps between what legacy systems provides and information management requirements must be solved without major disruptions and must provide low cost means of bridging the gaps and disconnects. To identify gaps in the collaborative environment infrastructure, models can be used to identify network connectivity, security and access services, tools, and databases, and most importantly the capability to interchange data between tools and databases. For example, does a user that needs to access data from multiple databases have to log on to each of the databases; if so then productivity inefficiency occurs. At its most fundamental level our approach consists in separating what is needed from what exists to get a better understanding of the pinpoint issues that need to be addressed.

2.5 Using System Engineering Enablers

To apply system engineering methodology to a product development enterprise we looked for tools, processes, methodologies, architecture frameworks, and any other enablers that could assist with the collaborative infrastructure gap analysis and help fill the gaps.

2.5.1 Architecture Frameworks

Our approach to product development is model centric. DoDAF provides a good methodology to perform a systematic analysis of information needs for specific product development processes. The result of the analysis has been used to produce an operations model that can be mapped onto the information infrastructure used in implementation of the development processes. DoDAF has proven extremely useful in helping frame the problem and collect the data needed to develop enterprise models. DoDAF originated as an architecture framework to analyze defense system-of-systems. Product development is a system of systems in the same sense and has similar problems with integration that occur with weapons systems integration. Not only is product development a systems of systems endeavor, but the fog of product development often rivals the fog of war. Graves, Hollenbach & Barnhart (2003) initiated the use of DoDAF for describing and analyzing product development and information systems.

2.5.2 Modeling Tools

UML and more recently SysML are system engineering enablers in that these modeling languages can be used to model the product development enterprise operation as well as model the product under development. The precision of a formal modeling language and tools enables identification of issues in the enterprise model that would otherwise not be found. We make extensive use of UML for product development enterprise modeling. UML with its formal syntax checking enables a precision not possible with text and view graphs. While UML is an effective modeling tool for product development which we use extensively, it doesn't provide a methodology for collecting the information needed to develop enterprise models. DoDAF provides a methodology to collect the information needed to build models.

2.5.3 Standards for Data Models and Data Interchange

A tool produces data in one format and other tools consume data in their own formats. In many cases data transfer between tools is achieved through specific point-to-point translations. The issue for implementation of the collaborative environment is where and when more general solutions are needed or are more cost effective. System Engineering Data Representation and Exchange (SEDRES 2) and now ISO 10303 AP233 has the objective of integrating systems engineering data into an overall product data management schema that achieves interoperability through the development of interchange standards for overlapping data between different classes of tools. The AP233 objective of integration of system engineering data into an overall data management schema is exactly what is needed. Tool vendors have been reluctant to adopt AP233 standards. However, recently AP233 are being translated into XML schemas and tools are providing XML output. These results make it easier to provide translations and mappings between tool formats and so AP233 is expected to play a more significant role in the future.

2.5.4 Appropriate use of enablers

No tool is a silver bullet and any tool can be used ineffectively and inappropriately. If, for example, DoDAF is seen primarily as a tool to engineer workflows then the tool use is at best misguided and at worst damaging. DoDAF does not in itself push architects toward viewing architecture as a static goal. Of course, once people get some new concept or tool, they are often unable to use it well and their brain hardens in some particular way. However, the way that we have used DoDAF is as a tool for information collection and analysis. Of course the requirements may change and if so the analysis changes. The fact that things change doesn't preclude trying to understand what is going on at a given time.

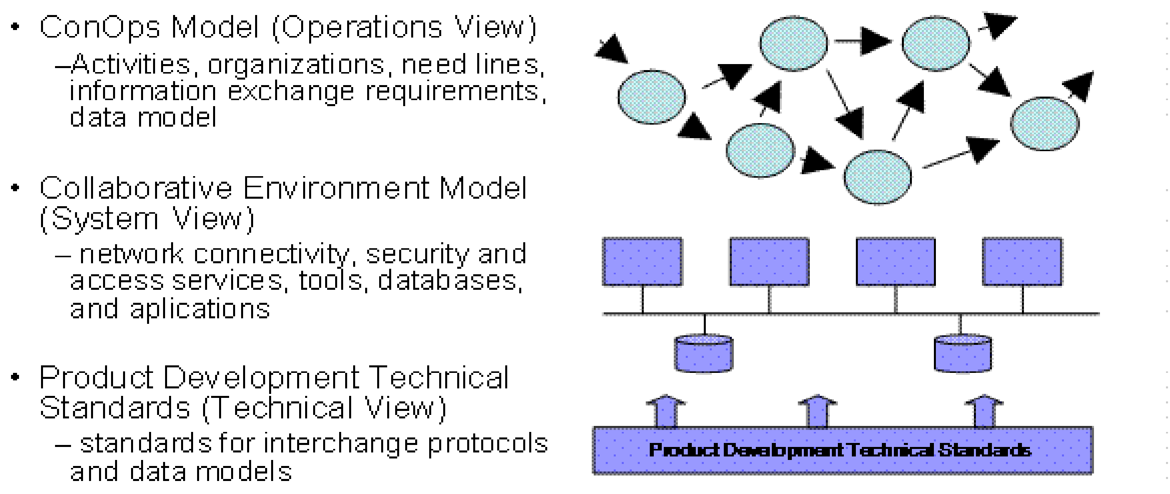
3 Product Development Architecture Views

A premise of this paper is that many causes of schedule slippage, loss of productivity and technical integrity in product development center on impedance mismatches between the operational requirements and collaborative information technology infrastructure that realizes the requirements. The mismatches are due to insufficiently detailed analysis of product development operational requirements, specifically analysis of the information required by each process activity to work effectively. The result of the incompleteness of the operational description is incomplete requirements for information technology. Consequently, the role of the information technology in providing an effective process implementation is often under appreciated. Good design decisions depend on good data and making decisions with poor data integrity leads to rework. Yet, the belief persists that tools will take care of the productivity problem. It is sometimes thought that baselineing product data in a configuration management system and requiring approval to change the baseline is sufficient strategy for data management. However, no coherent baseline can be established unless detailed descriptive information about the data is available to determine if the configuration managed data is consistent and coherent.

3.1 Three Architecture Views

Analysis of product development enterprise operations suggests the need for distinguishing between an operational model of product development and the people, tools, and information technology used to implement the development process. We refer to the people, tools, and information technology as the collaborative environment. The data interchange standards and communication protocols are a critical link between the operational model and the physical infrastructure. The availability of data interchange standards and tools that produce and consume data in standard formats affect the system architecture solutions that are potentially available.

The analysis of the product development suggests that an operations, or logical, model, a system, or physical, model, and the mappings between these models are needed to develop the operational requirements, and perform the gap analysis of the collaborative environment infrastructure's ability to support information requirements. **Figure 5** is an adaptation of a figure used in the ISO/STEP AP233 literature to describe the relationship between the systems engineering process, the supporting environment, and the systems engineering data model. This figure illustrates an operations view, a system view, and technical view that provide the basis for mapping between the operations and system view.



- ConOps Model (Operations View)
  - –Activities, organizations, need lines, information exchange requirements, data model
- Collaborative Environment Model (System View)
  - – network connectivity, security and access services, tools, databases, and aplications
- Product Development Technical Standards (Technical View)
  - – standards for interchange protocols and data models

Figure 5: Three Architecture Views

3.2 DoDAF

The DoDAF uses three views to represent the architecture of a system: an Operations View (OV), a System View (SV), and a Technical View (TV). Each architecture view has a number of products to describe the view. For product development the Operations View is the concept of operations. The Systems View is the collaborative environment used to realize the concept of operations, including network infrastructure, tools, and databases. The Technical View is the collection of standards for information exchange, including communications protocols and data exchange formats. **Figure 6** illustrates the three DoDAF views and their interrelationships as applied to product development. The Operational View is realized by the System View using the Technical Standards view. The flow is bidirectional in that systems technology and technical standards can affect the operational architecture and conversely. The System View requirements are derived primarily from the Operations View.

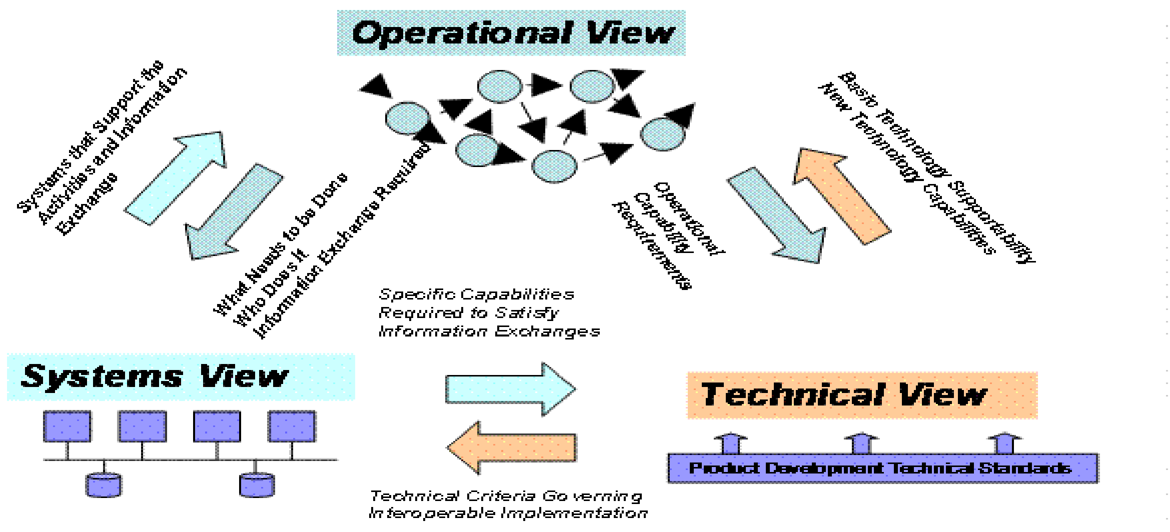

Operational View
Systems that Support the Activities and Information Exchange
What Needs to be Done
Who Does It
Information Exchange Required
Operational Capability Requirements
Basic Technology Supportability
New Technology Capabilities
Specific Capabilities Required to Satisfy Information Exchanges
Systems View
Technical View
Product Development Technical Standards
Technical Criteria Governing Interoperable Implementation

Figure 6: The DoDAF Architecture Views

The DoDAF diagram illustrates how these views affect each other. The effectiveness of a program is defined in terms of the effectiveness of its operations. The effectiveness of development operations depends on the collaborative environment (system) that realizes the operations.

3.3 Operations Architecture

Most aerospace programs produce a concept of operations, a system engineering master plan, other program plans and directives, and instructions to guide operation of the program. These documents constitute an operations architecture view in the DoDAF sense. However, these constituent operations documents often do not contain sufficient detail in their activity models, information exchange requirements, and data model to determine the collaborative environment (system architecture) requirements. In this section we describe how DoDAF operations architecture component views can be used to obtain the level of detail needed to establish system architecture (collaborative environment) and technical architecture requirements.

A product development program's policies, procedures, and concepts of operations documents are a natural starting point to construct a number of the DoDAF operational architecture products. These products help identify sufficient detail to identify the data producers, and to match consumers and producers with need lines and use this information to construct a logical data model that provides requirements for the collaborative environment implementation.

3.3.1 Context diagram for product data

The top level system engineering process that serves as the context diagram for the application of DoDAF to product development is represented in **Figure 7** as a "V" diagram with two dimensions. Time proceeds horizontally from left to right. The vertical dimension gives a stratification of tiers of organization and product decomposition. A "V" diagram, as is well understood, is a fiction; it does not show feedback cycles within the "V," and does not show that each major design cycle is its own "V." The diagram does show horizontal swim lanes that reflect organizational and product decomposition hierarchies. Information at a given level of detail flows within the swim lanes. Specifications flow down across the swim lanes and with test and integration results flow up across the lanes. The dotted lines correspond to milestone events. The transparent ovals are three areas where modeling and simulation plays a large role and which will be used for illustration in this paper. Simulation for requirements analysis uses a characterization of the product. Over time the requirements characterization evolves into a high fidelity product representation that mirrors the physical product as it is built. Verification requires a high fidelity product representation as a baseline product description.. Even after the air system is built, simulation plays a major role in product verification. The critical issue for the Verification is the ability to trace the linage of the component representations to the integration and test process. For this reason we focus on how model information flows up from Integration and Test to System Verification and Test.

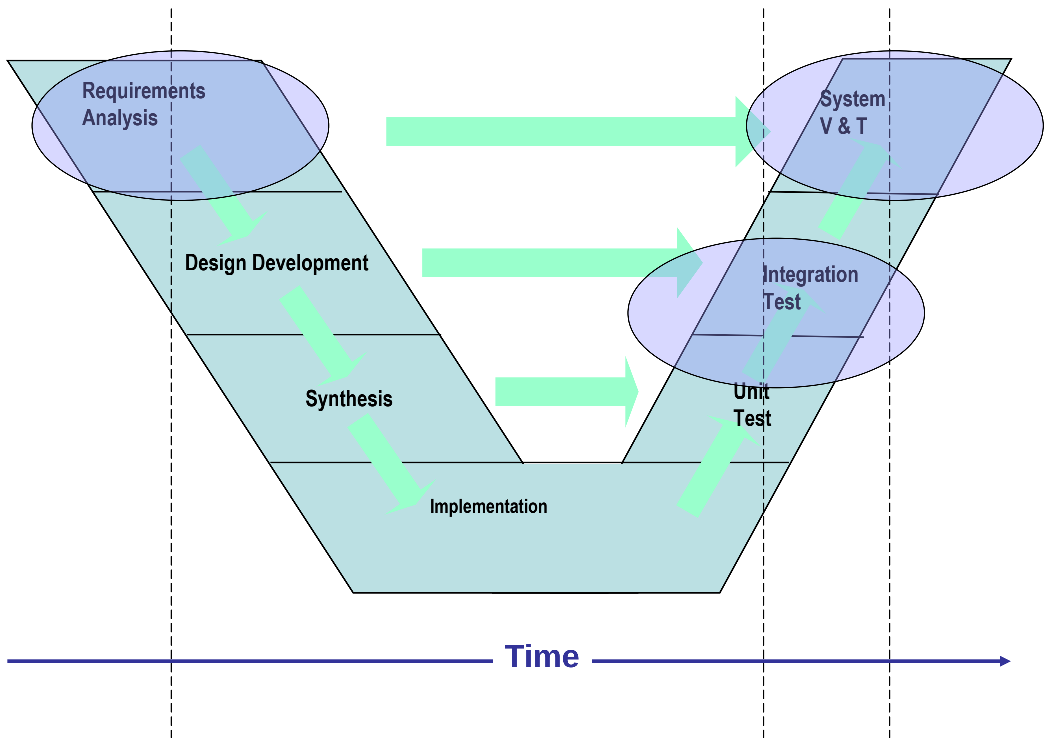


Figure 7: Top Level System Engineering Process

While emblematically we have used a "V" diagram to describe the product development operations much more detailed information is needed. Most ConOps recognize that teams need to exchange data and may even identify handoff agreements between teams. Program plans identify activities that take place to accomplish tasks, the performing organization, need lines, and time required to accomplish tasks. What is often missing in the ConOps is adequate provision for synchronizing data produced by concurrent activity and keeping the project information coherent. Design teams do not work in a vacuum. One team's results depend on results from other teams. Effective horizontal integration of a product development enterprise requires not only the configuration management of final releases of design data to manufacturing, but the sharing of data between product teams before final release. All of this data must be related and combined to achieve horizontal integration and enable collaborative product engineering. Evidence of non-recognition and inability to address concurrence can often be found in the approach to work flows. The work flows express a linear mentality; data can not be released until a chain of signoffs have been made, yet no provision has been made for the fact that data is dependent on access to other data. When such work flows are rigidly enforced one often finds that signoffs do not correspond to any completeness, usability, or quality metrics for the data being signed off on.

3.3.2 DoDAF Operations View

Figure 8 illustrates some of the DoDAF operational architecture view products and their interrelationships. Many of these components are identifiable as standard products produced by a program. At the top of **Figure 8** is a high level operational graphic (OV-1). We have used a Simulation Based Acquisition (SBA) diagram as a notional top level operations concept that determines organizations and roles. OV-5 is used to determine activities of organizations and their time lines. We have used a traditional "V-diagram" as emblematic of the operational activity model (OV-5). The Operational Node Connectivity Description (OV-2) describes exchange information needs of specific organizations within the enterprise. The Operational Information Exchange Matrix (OV-3) is used to describe the information exchanges between organizational nodes. The logical data model (OV-7) is a model of the kinds of product data required to accomplish development tasks. As we will see the DoDAF Logical Data Model corresponds to our information model. **Figure 8** illustrates the logical data model (OV-7) as a collection of data categories that include requirements, analysis, manufacture, integration test and verification and a number of product representations.

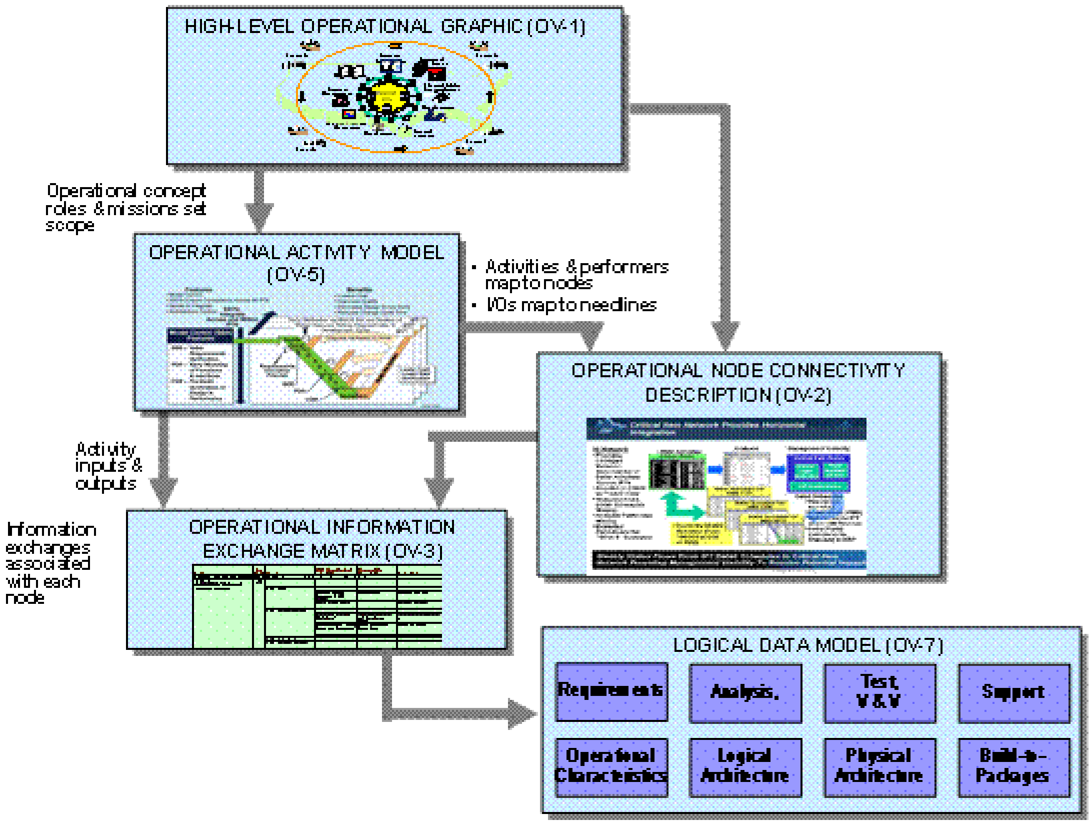

HIGH-LEVEL OPERATIONAL GRAPHIC (OV-1)
Operational concept roles & missions set scope
OPERATIONAL ACTIVITY MODEL (OV-5)
• Activities & performers map to nodes
• I/Os map to needlines
OPERATIONAL NODE CONNECTIVITY DESCRIPTION (OV-2)
Activity inputs & outputs
Information exchanges associated with each node
OPERATIONAL INFORMATION EXCHANGE MATRIX (OV-3)
LOGICAL DATA MODEL (OV-7)
Requirements
Analysis,
Test, V & V
Support
Operational Characteristics
Logical Architecture
Physical Architecture
Build-to-Packages

Figure 8: Operational Architecture Products

Figure 8 shows an information flow relationship between the five operations view products in the diagram. While work to produce the products may be done concurrently the information flow gives a precedence relationship leading to the logical data model.

3.3.3 Operational Activity Model (OV-5)

The Operational Activity Model (OV-5) describes the operational activities normally conducted in the course of performing the tasks of the organizational nodes and the information exchanged between operational nodes. We developed a number of activity models (OV-5) by interviewing participants to understand how consumers obtained their data. **Figure 9** is an example of an OV-5 Operational Activity Model represented as a UML activity diagram. The diagram shows "swim lanes" corresponding to component suppliers, data producers, system integrators, data consumers, and IPT leads. The model was constructed by interviewing participants about how they did work and how they wanted to work. **Figure 9** does not include all of the messages that flow between the participants, but it does identify the product data that flows between the groups and identifies who needs to signoff to certify authenticity of data.

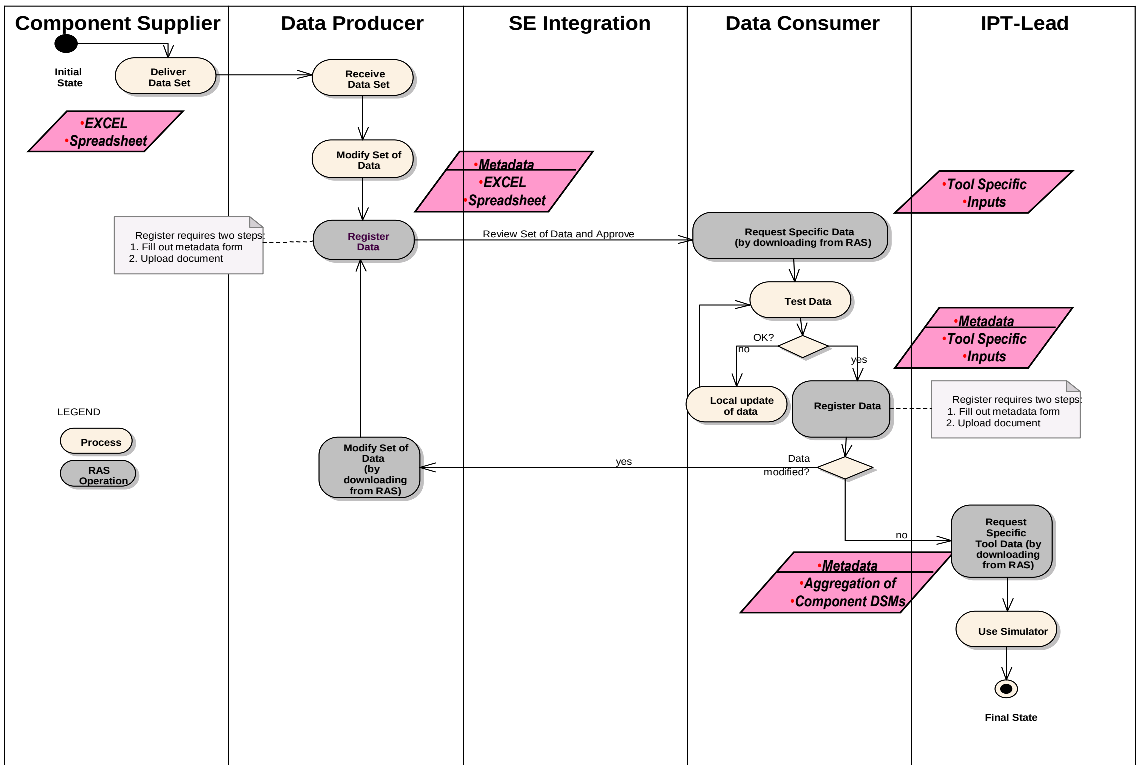

Component Supplier
Data Producer
SE Integration
Data Consumer
IPT-Lead
Initial State
Deliver Data Set
•EXCEL •Spreadsheet
Receive Data Set
Modify Set of Data
•Metadata •EXCEL •Spreadsheet
Register requires two steps: 1. Fill out metadata form 2. Upload document
Register Data
Review Set of Data and Approve
Request Specific Data (by downloading from RAS)
•Tool Specific •Inputs
Test Data
OK?
no
yes
Local update of data
Register Data
•Metadata •Tool Specific •Inputs
Register requires two steps: 1. Fill out metadata form 2. Upload document
LEGEND
Process
RAS Operation
Modify Set of Data (by downloading from RAS)
yes
Data modified?
no
Request Specific Tool Data (by downloading from RAS)
•Metadata •Aggregation of •Component DSMs
Use Simulator
Final State

Figure 9: Activity Model (OV-5) for Exchange between Producers and Consumers

A critical result of the activity analysis is determination of the interfaces between organizations. An interface description for an organization specifies what work products (data) flow between organizations.

3.3.4 Operational Node Connectivity Description (OV-2)

The Operational Node Connectivity Description (OV-2) tracks the need to exchange information between specific organizations within the enterprise to meet need lines. OV-2 is a model of the organizations and the information they exchange depicted as “need lines” with identifiers showing type of info and direction of flow. OV-2 is not a network diagram and does not show communication means. For product development the major milestones identify need lines and OV-2 serves the critical role for product planning in identifying data need lines for major milestone events. The data needed is determined by the success criteria for the event. The critical issue for program planning and assessment is to have the data required for a milestone event is sufficient detail to determine the risk of its availability.

After identifying organizations, organizational interfaces, and information flow (producing activity diagrams (OV-5) the next level of detail is to determine what specific data is required for a specific event. The example, used here for describing the role of OV-2 is analysis of the success criteria for the first manufactured Air Vehicle's first flight. **Figure 10** illustrates the integration and test flow leading to first flight. The lines in **Figure 10** do not distinguish between data flow, simulation connections, or hardware transfer from one organization to another. Of course for system analysis these distinctions have to be made very carefully.

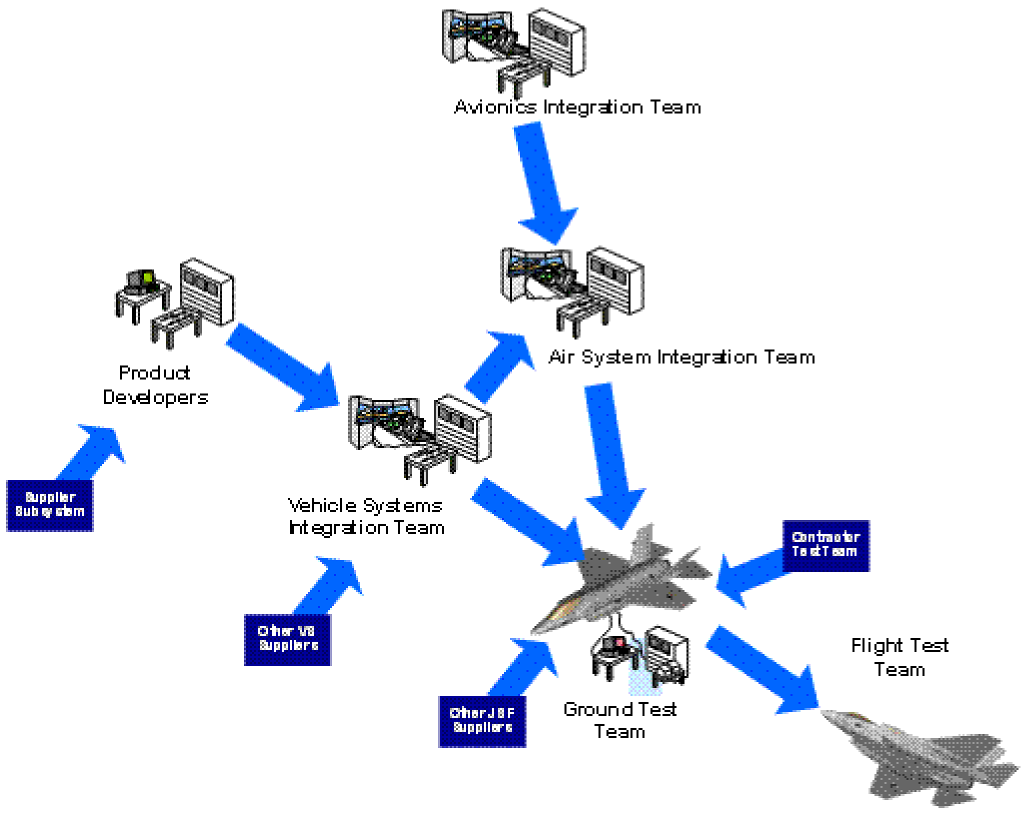


Figure 10: Information Flow Diagram that identifies M&S Need Lines for First Flight

The entrance criterion for first flight is a certification document that describes a specific flight profile. Certification requires that the aircraft and its subsystems and components have been qualified to meet the constraints imposed by the flight certification. Analysis of the requirements for component and system qualification can be used to determine the test activities needed for the first flight. Each activity has an entrance and exit criteria.

The result of this effort which identified the air system models needed for first flight is an Operational Node Connectivity Description (OV-2). The basis for constructing the models needed for first flight diagram **(Figure 10**) is work backward from first flight. For first flight all equipment on the air vehicle must be certified for flight. Models of the operating environment are used in ground test for certification and must be available at the ground test need line. Before ground test, integration models are used in labs for vehicle control systems and avionics which in turn require models from product developers.

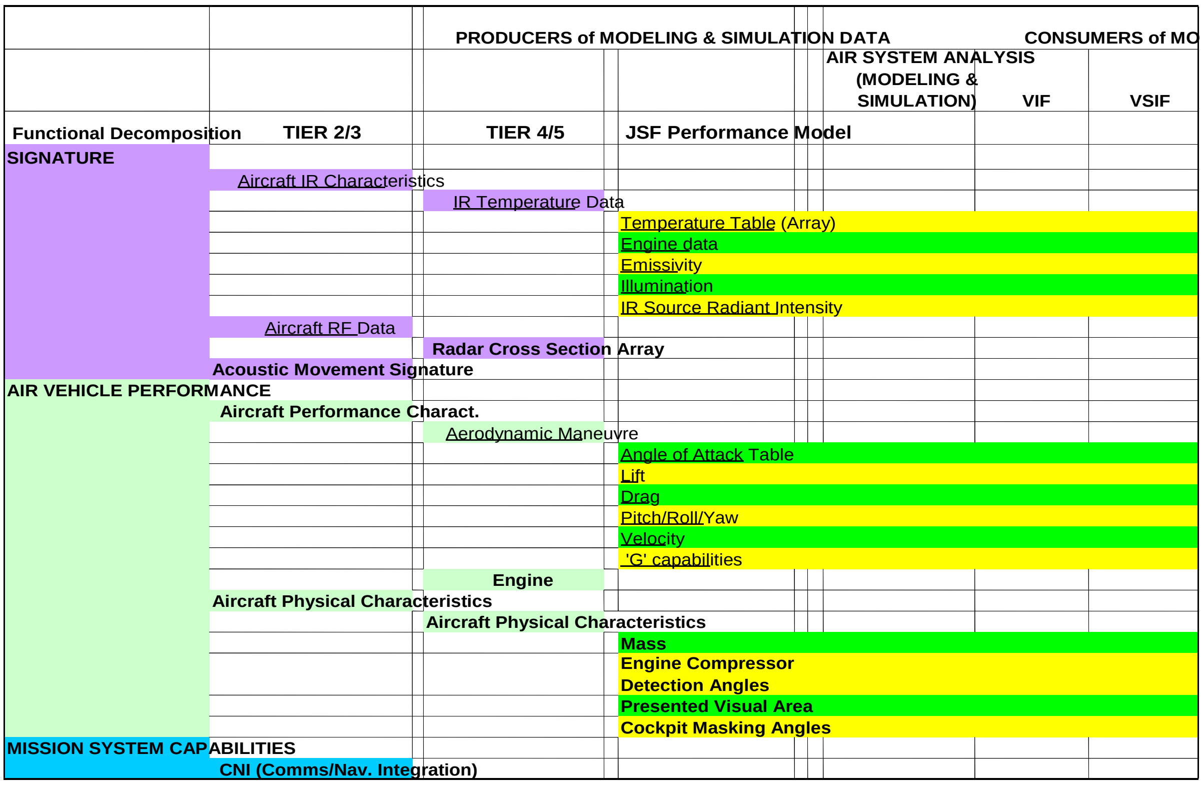
3.3.5 Operational Information Exchange Matrix (OV-3)

The Operational Information Exchange Matrix (OV-3) details information exchanges that identify "who exchanges what information, with whom, why the information is necessary, how the information exchanges must occur, and *when* it occurs. The purpose of an OV-3 diagram is to document the relationship across three basic elements of the Operational View: operational activities, operational nodes, and information flow. The operational activity model (OV-5) and the operational node connectivity model (OV-2) lead directly to identification of the operational information exchange matrix (OV-3). The focus is on specific aspects of the information flow. Each Information Exchange is related to an operational activity from OV-5 that produces or consumes it. We have found several kinds of diagrams useful for representing OV-3 information.

Constructing an accurate and complete OV-3 is likely too difficult and time consuming a task for an entire product development program. We have used OV-3 diagrams to determine what data was required for specific events and tasks. To construct an OV-3 we have used questionnaires that were filled out by data producers and consumers. The questionnaires for data producers are used to determine what product data they are producing, in what form, and who their customers were and where it is stored. From data consumers we use spread sheets to determine what data they need for specific design, integration and test, and verification tasks. The resulting OV-3 diagrams can be represented using spreadsheets using column headings for information producer, consumer, content, task it supports, triggering event, communication medium, format (include any standard being followed), periodicity, time to transform information into consuming tool's format, and classification or confidentially.

3.3.5.1 Information Exchange for Performance Models

Figure 11 is an example of an Information Exchange Matrix for Models (OV-3) that we have constructed for product modeling and simulation. The matrix identifies the consumers of these models and the producer of each model. This diagram is organized by a functional decomposition hierarchy. At each node in the functional decomposition we define a performance model by a collection of parameters. As each of these models must trace to engineering and test data we need to identify the model producers. **Figure 11** OV-3 diagram describes this representation together with the sources for each component model. The first three columns identify the levels in a product decomposition used to identify the models. Column four identifies the model producer. Columns to the right identify the labs that use the models and their need lines.

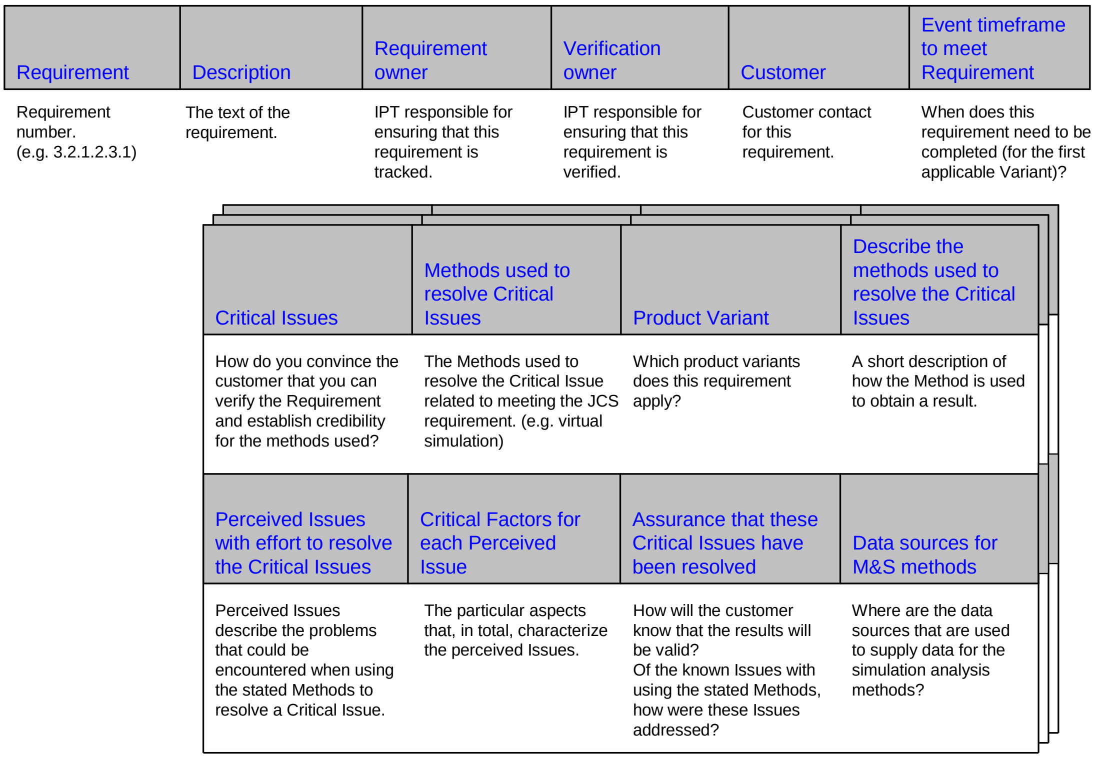


		PRODUCERS of MODELING & SIMULATION DATA			CONSUMERS of MO
			AIR SYSTEM ANALYSIS (MODELING & SIMULATION)	VIF	VSIF
Functional Decomposition	TIER 2/3	TIER 4/5	JSF Performance Model		
SIGNATURE					
	Aircraft IR Characteristics				
		IR Temperature Data			
			Temperature Table (Array)		
			Engine data		
			Emissivity		
			Illumination		
			IR Source Radiant Intensity		
	Aircraft RF Data				
		Radar Cross Section Array			
	Acoustic Movement Signature				
AIR VEHICLE PERFORMANCE					
	Aircraft Performance Charact.				
		Aerodynamic Maneuvre			
			Angle of Attack Table		
			Lift		
			Drag		
			Pitch/Roll/Yaw		
			Velocity		
			'G' capabilities		
		Engine			
	Aircraft Physical Characteristics				
		Aircraft Physical Characteristics			
			Mass		
			Engine Compressor		
			Detection Angles		
			Presented Visual Area		
			Cockpit Masking Angles		
MISSION SYSTEM CAPABILITIES					
	CNI (Comms/Nav. Integration)				

Figure 11: Information Exchange Matrix (OV-3) Organized by Models, Producers, & Consumers

3.3.5.2 Verification Information Exchange Matrix

When any verification, assessment, or integration process can trace its results to a digital representation of the product, the product representation is complete for that stage of development. To determine completeness criteria we are developing Information Exchange Matrices (OV-3) tailored for verification. **Figure 12** is an example a verification Information Exchange matrix. The matrix is to determine how models are used in verification, when they are needed, what level of detail and fidelity is required, and what is the authoritative source of the models. The verification information exchange matrix (OV-3) can be used to identify risks associated with modelling and simulation. **Figure 12** organizes information by requirement. The top grey row identifies the requirement by name, description, owner of the requirement and the owner of the verification responsibility.



Requirement	Description	Requirement owner	Verification owner	Customer	Event timeframe to meet Requirement
Requirement number. (e.g. 3.2.1.2.3.1)	The text of the requirement.	IPT responsible for ensuring that this requirement is tracked.	IPT responsible for ensuring that this requirement is verified.	Customer contact for this requirement.	When does this requirement need to be completed (for the first applicable Variant)?

Critical Issues	Methods used to resolve Critical Issues	Product Variant	Describe the methods used to resolve the Critical Issues
How do you convince the customer that you can verify the Requirement and establish credibility for the methods used?	The Methods used to resolve the Critical Issue related to meeting the JCS requirement. (e.g. virtual simulation)	Which product variants does this requirement apply?	A short description of how the Method is used to obtain a result.
Perceived Issues with effort to resolve the Critical Issues	**Critical Factors for each Perceived Issue**	**Assurance that these Critical Issues have been resolved**	**Data sources for M&S methods**
Perceived Issues describe the problems that could be encountered when using the stated Methods to resolve a Critical Issue.	The particular aspects that, in total, characterize the perceived Issues.	How will the customer know that the results will be valid? Of the known Issues with using the stated Methods, how were these Issues addressed?	Where are the data sources that are used to supply data for the simulation analysis methods?

Figure 12: Vérification Information Exchange Matrix (OV-3)

The diagram is organized by (the activity of) verification for top level requirements. The operational nodes, i.e., organizations are first the organization that owns the requirement and the organization that owns the verification of the requirement, the customer. The *when* is covered by the Event timeframe column heading. The information flow that is of interest in this diagram is information required by the verifying agent of the requirement; not all information, but the information relevant for critical issues regarding the verification.

What	- information relevant to critical issues regarding a req. verification
Whom	- primary organizational nodes and info suppliers to verifier
Why	- required to support critical issues in verification
How	- not explicitly stated on this form
When	- event timeframe

There are even other similarities with the DoDAF OV-3 template. For example, performance attributes and information assurance correspond to our column head "issues or risk with effort to resolve critical issues". Again let me repeat that much of our work is in designing forms to get the information needed to build a complete OV-3 or any other picture. When we get all of the information that we would like to have, then we could put it in a more standard template form.

3.3.6 Logical Data Model (OV-7)

The information management objective for product development is to (1) provide authoritative, accurate data for exchange between operational nodes, (2) to characterize product variants and configurations, and (3) to ensure integrity throughout the chains of data production from primary engineering data sources. Achieving the management objective requires identifying exactly what kinds of data need to be exchanged between the organizational nodes, including the data form, content, and how data depends on and is related to other data. The kinds of data, the data interrelationships, the form and content, and how the data relates to the product variants and configurations constitute a model of the data, the product, and the organizations and processes that produce the data. The DoDAF Logical Data Model (OV-7) represents the kinds of data exchanged in the system operations.

3.3.6.1 The Role of the Logical Data Model for product development

The purpose of OV-7 in product development is to define the scope and content of the information to be managed, insure that the data is managed, and that there are standards for data display and interchange. OV-7 is at the interface between the DoDAF Technical Architecture View and the System Architecture View. OV-7 places requirements and constraints on the data authoring and management tools. The logical data model defines what information is needed to manage and perform technical tasks on a program. If the information management system can not manage this information then the program will surely be in trouble. The existence of a good data model and the ability of the information management system to implement the data model is a good predictive measure of program effectiveness.

For the operational architecture to be implemented by the system architecture there must be tools that produce and manage this collection of data. The choice of tools and data management systems is often made before a program begins and the program’s data interchange requirement are well understood. The result is that tool’s import and export formats influence the choice of interchange formats in the Technical Architecture.

3.3.6.2 The scope of the data model

In product development the scope of information to be managed includes work products produced by requirements analysis, design synthesis, manufacturing, integration and test, verification, and support activities. The scope is bounded by what information might be needed to provide traceability for verification, and to understand consequences of design changes and modifications. **Figure 13** illustrates some of the kinds of data exchanged by product teams during the development process. The kinds of data, or classes of data, include requirements, use cases, test cases, component designs, and test results. Members of these classes of data are both produced and consumed by organizational nodes such as Integrated Product Teams (IPTs). An organizational node responsible for development of a product component takes as input data about the component and outputs data about the component to higher level IPT nodes and outputs data about subcomponents to lower level IPTs. An OV-7 for product development includes organizational structures and data associated with nodes in organizational structures as well as product representations, templates for data produced by the development process such as requirements analysis, systems analysis, design, manufacture, integration results, and verification and test results, and data signifying that program milestone events are completed.

A significant part of the data required for technical management is metadata about work products such as descriptions of who produced the data and how it was produced. This metadata is required to establish the authority and integrity of the data. Product data needs to be identified not only by class, but also by applicability conditions. An applicability condition identifies which classes of products, or individual products, the data item is applicable to. For example, the organizational node responsible for the fuel system identifies which variants of the vehicle specific design components are valid for. Note that the applicability references of data items are implicit in **Figure 13** which identifies what data is exchanged between nodes. The kind of analysis illustrated in **Figure 13** can be used to produce a data dictionary, and a model that shows containment, uses relationships, and other dependences between classes of data. Defining the content of the data classes requires analysis of the activities that produce and consume the work products.

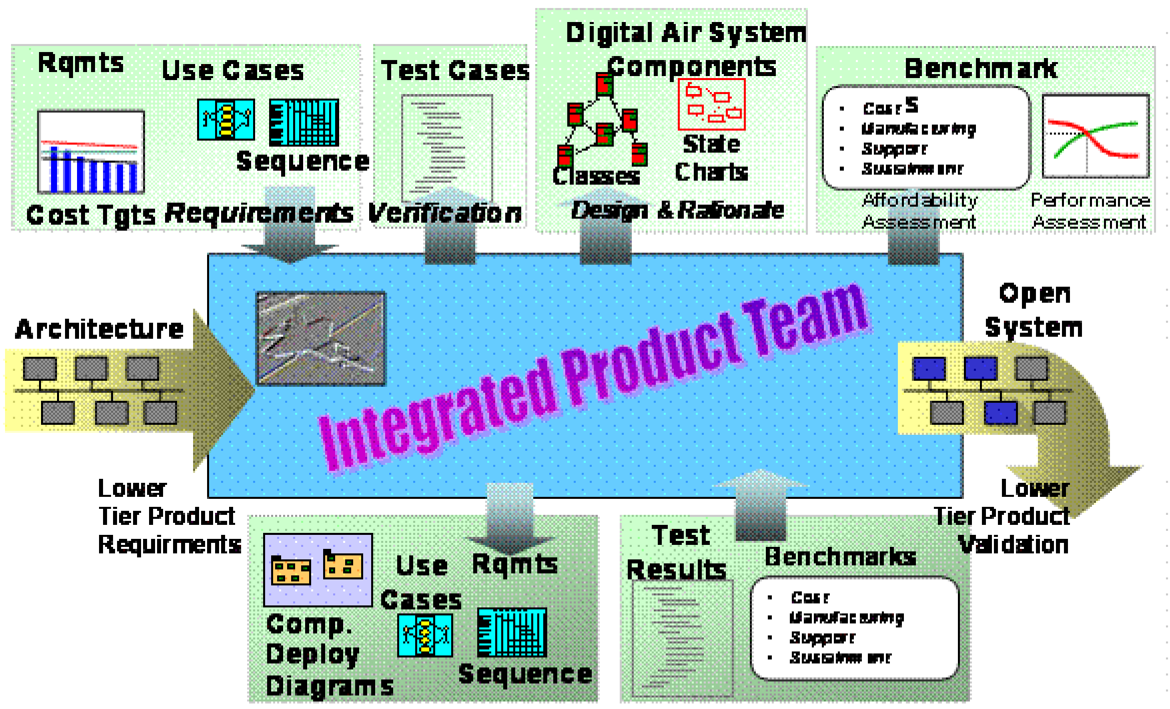


Figure 13: Information Flow in and out of an Integrated Product Team

3.3.6.3 A Product Development Logical Data Model

The construction of such a model is greatly facilitated by the development of the DoDAF operational view products. The operational information exchange matrix (OV-3) and the Operational Node Connectivity Description (OV-2) drive the construction of the logical data model (OV-7). Operational informational exchange matrix (OV-3) establishes the need to exchange information between different tools used in support of system engineering activities.

3.3.6.4 Representation of the Logical Data Model in a modeling language

OV-7 represents the kinds of data exchanged in systems operations. Many modeling languages can represent system engineering data concepts such as products, processes, work breakdown structures, events, and organizations as objects. We have used UML class models to represent product development logical data models. The UML class model is a good representation for communication between stakeholders. UML represent objects and classes. UML objects have attributes, and relationships between objects. Classes describe the collection of objects that have the same attributes. Each class defines a collection of instances which are members of the class. Subclasses of a class are defined by constraining attribute values to be more restrictive. As will be noted in the Technical Architecture section the UML diagrams are generated from XML schemas. The XML schemas are the master representation and define the data interchange formats and can be used by a Web-based access system. A fuller discussion of these topics may be found in Graves, H. & Johnson, C. *et al* (2004).

In developing a class model for a product development logical data model we have used the concept of an information resource to describe individual configuration-managed items. For data management the critical question is what product entity the information resource applies to. We associate information resource classes with product decomposition classes. An information resource is associated with the product decomposition objects to which it applies. The Product Decomposition class is shown as having associated information resources. Each information resource further has attributes that identify the source of the item and context of use. **Figure 14** is a top level representation of a UML class model for product development. The diagram shows a top level class, entity, and two classes inheriting from entity, navigation node and information resource.

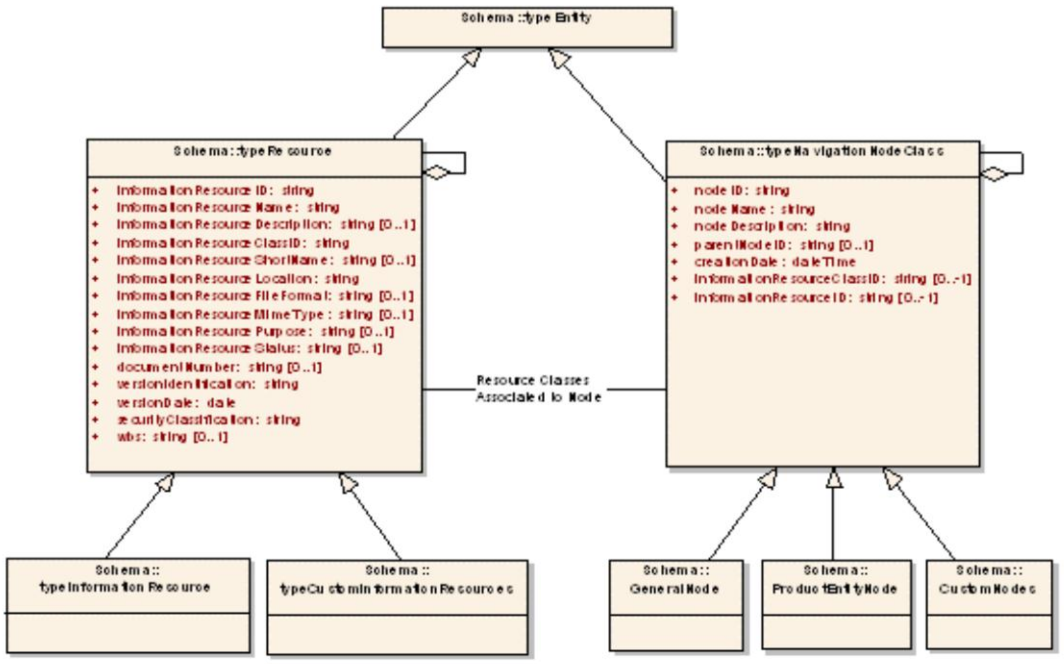

Schema::typeEntity
Schema::typeResource
+ InformationResourceID: string
+ InformationResourceName: string
+ InformationResourceDescription: string [0..1]
+ InformationResourceClassID: string
+ InformationResourceShortName: string [0..1]
+ InformationResourceLocation: string
+ InformationResourceFileFormat: string [0..1]
+ InformationResourceMimeType: string [0..1]
+ InformationResourcePurpose: string [0..1]
+ InformationResourceStatus: string [0..1]
+ documentNumber: string [0..1]
+ versionIdentification: string
+ versionDate: date
+ securityClassification: string
+ wbs: string [0..1]
Resource Classes Associated to Node
Schema::typeNavigationNodeClass
+ nodeID: string
+ nodeName: string
+ nodeDescription: string
+ parentNodeID: string [0..1]
+ creationDate: dateTime
+ informationResourceClassID: string [0..-1]
+ informationResourceID: string [0..-1]
Schema:: typeInformationResource
Schema:: typeCustomInformationResources
Schema:: GeneralNode
Schema:: ProductEntityNode
Schema:: CustomNodes

Figure 14: A Top Level Product Development Metamodel

Figure 15 is an elaboration of the class structure for a product. A product is described by decompositions such as the requirements, functional, logical, and physical decompositions in **figure 15**. The product class is shown as having associated decompositions. Product decomposition is a hierarchy of product decomposition objects. Thus, the product decomposition class is shown as having a child relation. Two product decompositions may share some objects and not others. For example, radar may occur in both a functional and a logical decomposition, but there may be no explicit radar object in the physical decomposition. Product decomposition classes are related by an identification relationship. Other relationships such as allocation and implementation connect objects of the different decomposition classes.

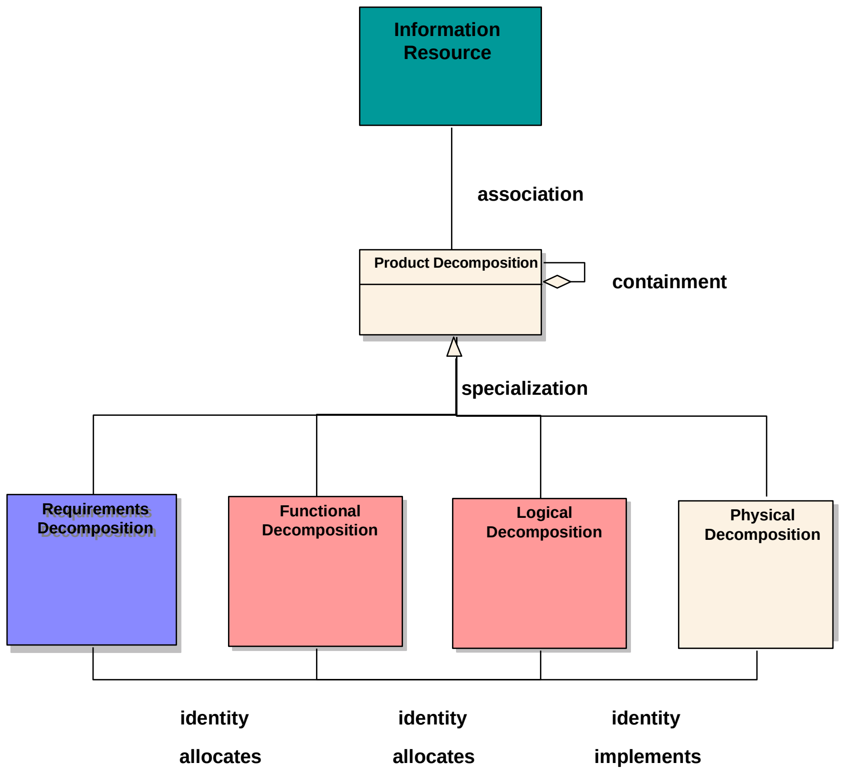


Figure 15: A Class Model for Product Decompositions

3.4 Systems Architecture

The success criteria for a product development system in its most simple form is that developers have access to the data that they need, when they need it, in the form that they need it, and have assurance that the data is accurate. In addition privacy and security criteria dictate that data is only available to individuals with authorized access. For product development the adverse impact of the information technology part of the collaborative environment failing to meet the success criteria is often underestimated. Product developers reason that they can do their jobs and disconnects in the information technology infrastructure will only impact their work with minor inefficiencies. “If I need data that I don’t have, I can simply call or email the right person and get it.” The real situation is often more like a pilot flying on instruments where the instruments are giving incomplete and perhaps even inaccurate data. Product development for complex systems has no alternative to instrument flying. While one may know the person who owns the data required finding the right individual is not always easy. Even more serious is that since data is generally dependent on someone else's data and often the owner may not know that the sources his data depends on may have changed, the result is that there is no accurate, coherent baseline for product data. How does this situation arise and what can be done about it? At the heart the problem is caused by an ill defined logical data model which then has an ill defined physical realization. The physical realization of the logical model does not account for the fact that data is produced by multiple tools in inconsistent formats with overlapping content. Consistency checking is primarily a manual operation. Requiring data to be centrally configuration managed does not in itself guarantee that the data is consistent. A good operations model should determine access and query requirements for the system model. Often the issue with satisfying data model requirements stems from not adequately capturing the requirements.

The DoDAF Systems Architecture describes the networks, facilities, tools, and functions that perform the information exchange and processing which implement the Operations Views. The primary reason for analysis and modeling of the collaborative environment is to determine if it can be used to implement the operations architecture in an effective way. The Systems Architecture can be used to collect the information needed to construct the product development system model (collaborative environment). The Systems View differs from the Operations View in the systems products focus on specific, physical systems. DoDAF specifies a number of products for the Systems View. For product development the Systems Architecture is the program's collaborative environment.

3.4.1 The information stack

The areas of concern for system architecture are naturally layered: with network connectivity on the bottom, then data transfer and transaction protocols, and finally semantic interoperability on top. **Figure 16** illustrates this stack. The most difficult and unique interoperability issues for product development are semantic interoperability and our examples will focus on this. The semantic interoperability issues are the ability to implement a data model sufficient to support the operational information needs of the organizational nodes. For the most part mature organizations have the network connectivity and ability to share data between nodes. The one place where this capability is sometimes problematic is in providing controlled access where security and control of access privileges make data sharing difficult.

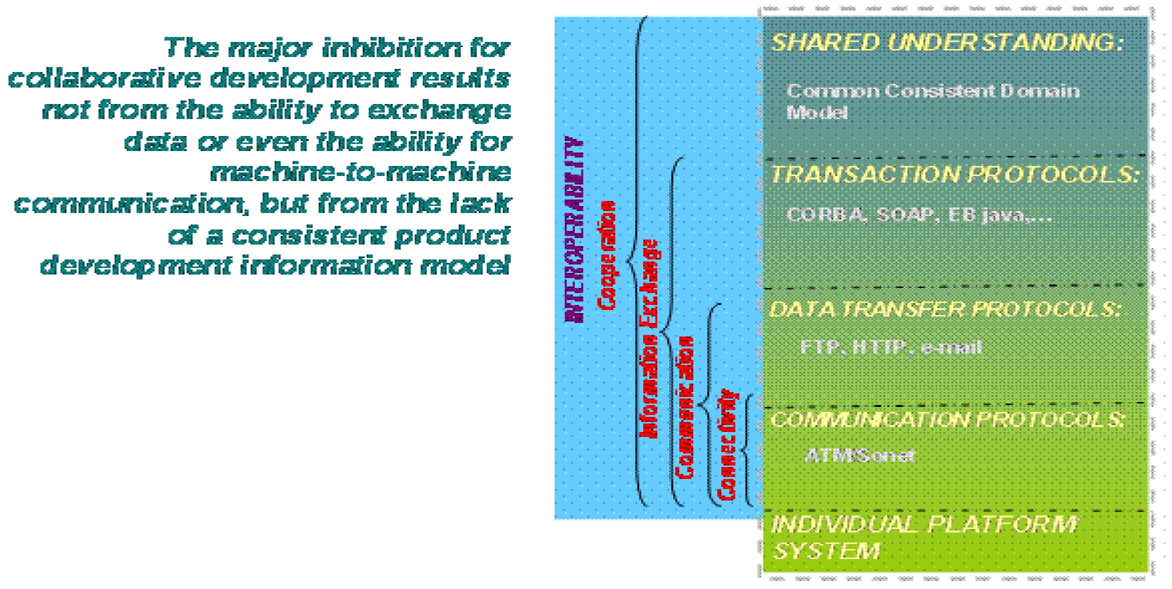

The major inhibition for collaborative development results not from the ability to exchange data or even the ability for machine-to-machine communication, but from the lack of a consistent product development information model

Figure 16: The System Information Stack

3.4.2 Integration Testing Examples

The examples on System Architecture in this paper focus mainly on labs used for integration testing. Integration testing spans the product development process after design and manufacture and before verification. System components are tested as units, integrated into larger assemblies which are tested, leading to integration and testing of the end product. In modern product development modelling and simulation play a significant role in this phase of product development. System components are often designed and developed as a series of models before the components are physically constructed. As a component is modelled and then constructed it is tested in integration labs through operation within simulated environments. A simulated environment contains models of other product components and models of the component under test's operating environment.

The systems view for Integration testing describes the equipment, network interconnections, and the functionality used for integration testing. The DoDAF System views help in analysing the maturity of the integration and test facilities to perform their job and help with measuring progress toward completion of integration testing activities. **Figure 17** illustrates a systems view of the labs involved in integration and testing and verification for an air system.

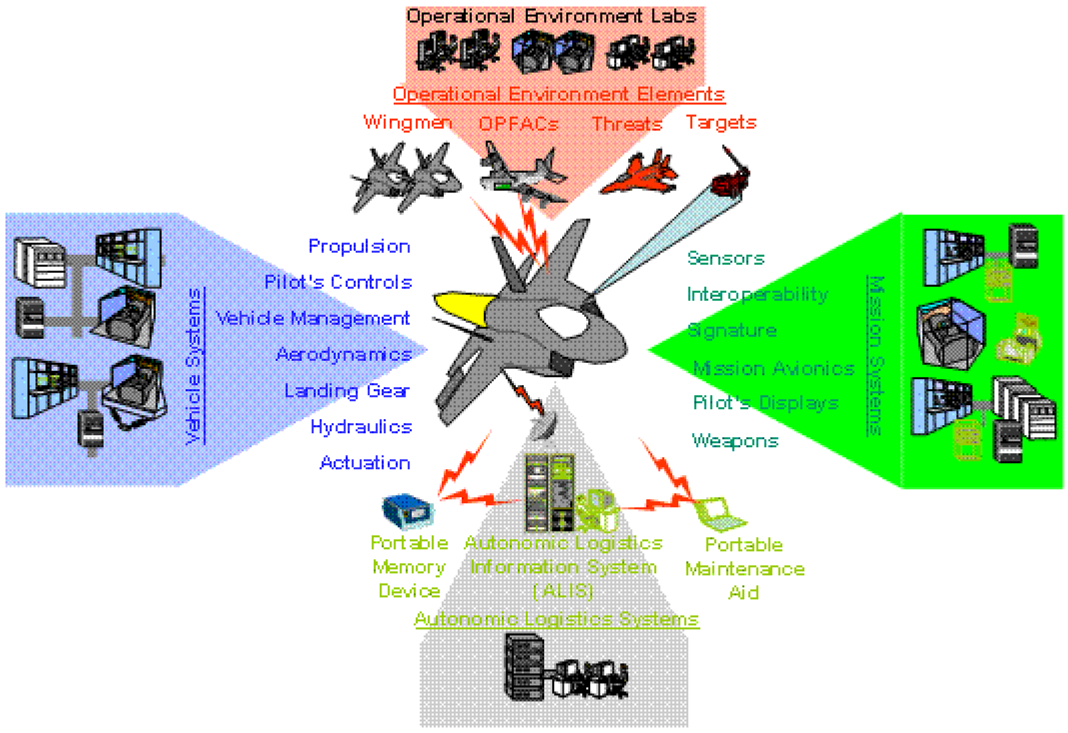


Figure 17: Labs Used For Integration Testing

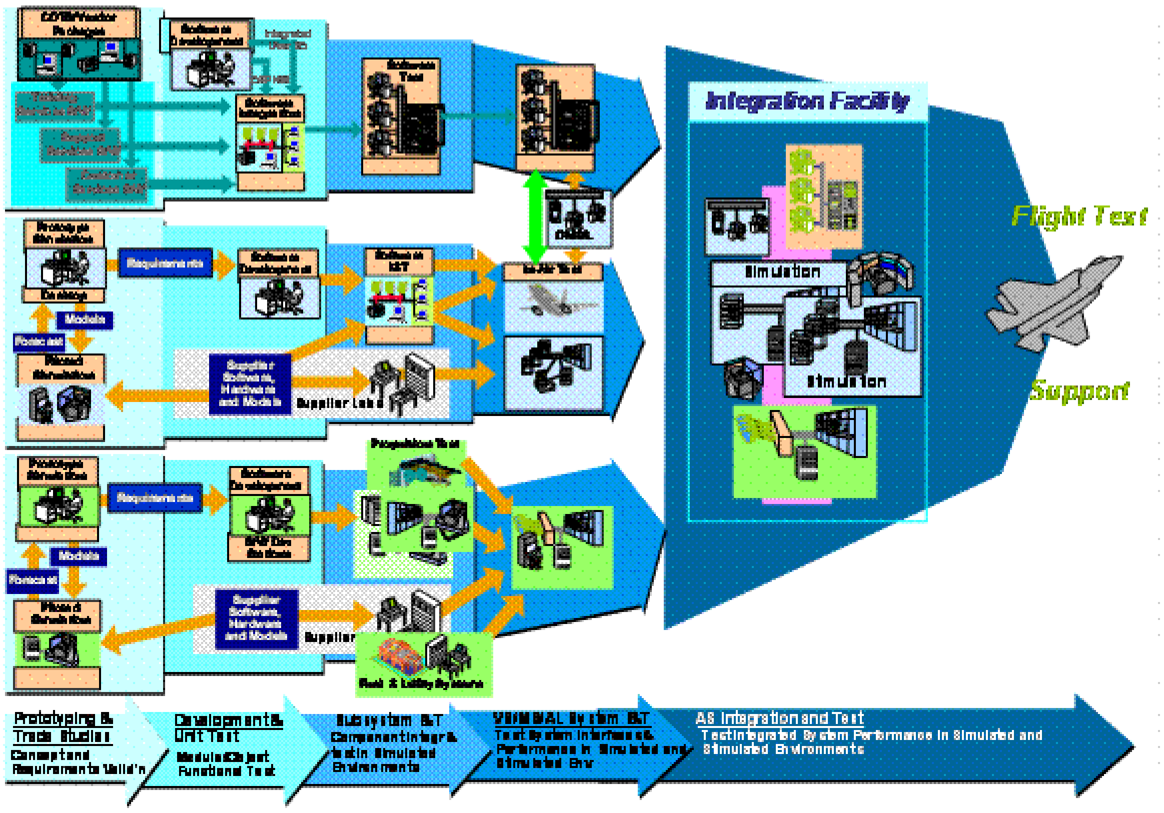
The Systems View of the labs describes the equipment, network interconnections, and the functionality used for simulation tasks. **Figure 17** illustrates the collection of labs used for integration testing as partitioned into mechanical, mission, operational, and support concerns that reflect basic product requirements.

3.4.2.1 Success Criteria vs. Lab Maturity

Integration testing maturity success criteria for an air system can be defined in terms of certification for flight. Certification for flight depends on qualification of components and integration and qualification of components into subsystems. Progress toward certification can be measured as a function of completion of qualifications leading to certification for flight. Qualifications are dependent on other qualifications and precedence relations can be used to define critical paths among the integration activities. The maturity of labs can be assessed in terms of their readiness to perform qualifications.

3.4.3 Systems Interface Description (SV-1)

The Systems Interface Description (SV-1) identifies systems nodes and the interfaces between the nodes. For product integration testing the nodes are the labs. There are several kinds of interfaces between the labs. Both data to initialize models, models and hardware are transferred or shared between labs and there may be distributed simulation interconnection. **Figure 18** is a schematic illustration of the nodes (labs) and interfaces between nodes required for the test and integration phase of product development.



COTS/Vendor Packages
Requirements
Models
Forecast
Supplier Software, Hardware and Models
Supplier Labs
Propulsion Test
Fuel & Utility Systems
Integration Facility
Simulation
Flight Test
Support
Prototyping & Trade Studies
Conceptand Requirements Valid'n
Development & Unit Test
Module/Object Functional Test
Subsystem I&T
Componentintegr & testin Simulated Environments
VS/MSAL System I&T
Test System Interfaces & Performance in Simulated and Simulated Env
AS Integration and Test
Test Integrated System Performance in Simulated and Stimulated Environments

Figure 18: Identification of Labs and interfaces between the Labs

The sequence of overlapping arrows at the bottom of the diagram illustrate critical milestones (need lines) starting with prototyping and trade studies and ending with flight test support. The block arrows above form a tree with root at flight test support. The activities of multiple labs flow through interfaces to support flight test. In **Figure 18** the yellow arrows connecting the block arrows represent both network connections for data flow and in some cases, interconnections between simulation systems.

3.4.4 Systems Functionality Description (SV-4)

Collaborative environment functionality focuses on the availability of product data to support design, assessment, and to provide traceability of results. The Systems Functionality Description (SV-4) product documents systems function hierarchies, and the data flows between them. The primary purpose of the Systems Functionality Description product is to (a) develop a description of the necessary information flows that are required by and output from each system, (b) ensure that the functional connectivity is complete, and (c) ensure that the functional decomposition is to an appropriate level of detail. The SV-4 product focuses on describing the flow of data among system functions and is the system view counterpart to the OV-5 Operational Activity Description.

Producing the data that a developer requires often involves accessing and combining data from multiple sources. **Figure 19** illustrates four organizational roles: administrator, producer, consumer, and examiner and some of the functions required for interaction with the collaborative environment.

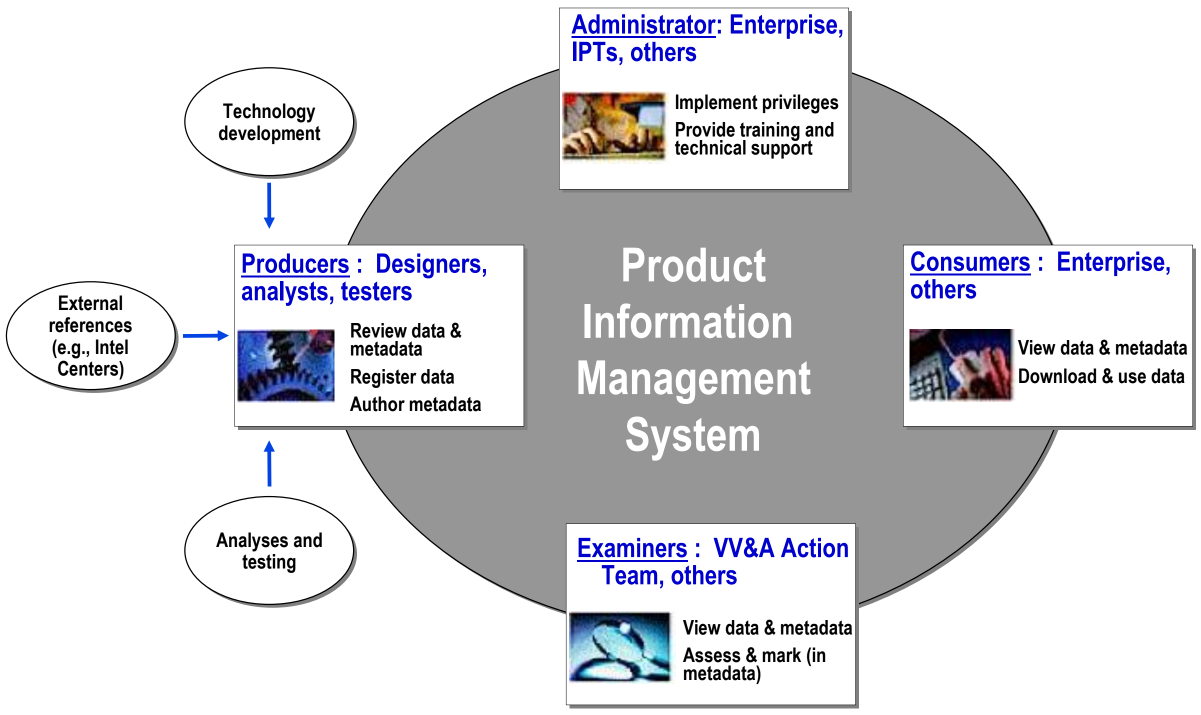

Administrator: Enterprise, IPTs, others
Implement privileges
Provide training and technical support
Technology development
External references (e.g., Intel Centers)
Producers : Designers, analysts, testers
Review data & metadata
Register data
Author metadata
Product Information Management System
Consumers : Enterprise, others
View data & metadata
Download & use data
Analyses and testing
Examiners : VV&A Action Team, others
View data & metadata
Assess & mark (in metadata)

Figure 19: System functionality for modelling and simulation data access and registration

The designer of a product subsystem needs visibility and access to all technical documents (test reports, analysis documents, specs…) relating to the system that the designer is working on. The full view of a particular subsystem is needed to determine potential impacts, when any of information changes. In addition, the designer needs visibility into requirements, design, build, and qualification documentation for each product configuration. At a program level a Critical Design Review (CDR) event the success criteria might be to have all of the build-to-packages and supplier produced component integration models complete. Later this information is used to status design and certification at key points in time, e.g. first flight. In addition to the producers, managers and control boards need to review and signoff on design and analysis results. The engineer charged with signing off on a result wants to know how it was produced, who produced it, what tools and processes were used to produce the result. The engineer may want to trace the linage of a result back to detailed engineering data. Data is authoritative if its context of applicability is identified and its source is identified as coming from the responsible organization. These are all requirements that the system architecture implementation must address.

3.5 Technical Architecture

The DoDAF Technical Architecture is the collection of technical standards used for interoperability at each of the different levels of the system interoperability stack (see **Figure 16**). The Technical Architecture provides the bridge between the Operations Model and the System Model in that the technical architecture provides a syntax that implements the data operations in the system model. For product development technical standards encompass interoperability at the transport level, the transaction level (authentication, registration of data), and at the application level for searching and formatting data. In our experience we have found transport standards are usually in place. As collaborative environments evolves to meet operational requirements for user authentication and single sign on for network and database access emerging standards will be adopted for these protocols. Much of our current work is focused on data interchange standards. Data interchange standards is the area addressed by AP233. Their approach has been to produce interchange standards and encourage tool vendors to support these standards. The success of this effort has been mixed and so the use of AP233 standards is limited by tool support.

3.5.1 Data Interchange Standards

Each tool used in the physical model produces or consumes data that is part of the logical data model. The existence of a well defined translation between data formats and semantics of data producing and consuming tools is evidence of maturity. For a translation to be well defined does not require that the translation be automated to be repeatable. According to the general system complexity argument point-to-point translations are more complex and hence costly to maintain than translation into a universal format with a translator from-to each specific tool format. In practice where point-to-point translations exist there may be no compelling business case to change to a universal format.

The data model illustrated in **Figure 13** specifies the scope of product information that is needed for interchange between tools and individual developers. The model is influenced by AP233; however, its primary concern is to reflect the actual products being developed on the particular program. As we have indicted much of the data to be managed that needs to be managed is metadata about authoring tool produced data files and documents such as release authorization notices that signify conclusion of events. Currently there are no standards for metadata or program technical documents. The data model described in **Figure 14** can be converted into XML schemas to provide data interchange standards. By making proper correspondences between UML class constructions and XML schema constructions modeling tools can automatically translate between UML class diagrams and XML schemas. Further, the UML data model is generated from XML schemas. Each class of the data model is represented by an XML schema. The class schema defines a collection of named class attributes and each class attribute has a value type. An XML file that is validated by the schema is an instance of the class. New classes can be defined by inheriting properties from other classes.

3.5.2 Web-based Standards

Our group has been a strong advocate of using Web-based standards for transaction protocols and data interchange standards. Where Web-based standards exist they satisfy the criteria for openness and Web technology lends itself to extensibility and flexibility. Web standards developed by industry and academics enable machine-to-machine communication, enabling a new level of computer-mediated automation. Web standards permit the evolution of the technical architecture while minimizing programming costs. Web standards use XML as the underlying machine-readable format. XML is a text-based language tailored for data representation that has extensive commercial support. XML works well for expressing data interchange standards. The emerging web service protocol standards work well for communication between system nodes. Web services use HTTP transport protocol and the Simple Object Access Protocol (SOAP) for transactions. SOAP provides a standard format for transaction requests. Other standards such as the Security Assertion Markup Language (SAML) are embeddable inside SOAP messages to represent specific kinds of transactions.

3.5.3 An Information Management Architecture

Commercial off the shelf (COTS) tools can be integrated to produce product development information management systems that make use of enterprise models. Web-technology provides the capability to integrate component product data management systems to satisfy information management requirements. An architecture for authoritative information system is described in Graves, Hollenbach & Barnhart (2003). **Figure 20** shows a Web-portal that is intermediary between the database systems where data is stored and the desktop web-browsers that host a Graphical User Interface (GUI). This architecture is configured with the Information Model shown in the light blue. Users register, access, and update data in the databases using the GUI. **Figure 20** shows the top level architecture of a Web-based system to register and access product data. The architecture uses a model to organize product data. The model is used to configure the system and provide a mechanism to locate and access product data. The configuration of the information access system includes not only the model that provides the organization for the data, but where the data is actually located and how to access the data from a specific repository. The model contains entities that represent products and product components as well as organizations, tools, and processes. Data is associated with the entities in the model. Product data is updated on a daily basis. The model is updated less frequently, but it is updated as the design structure changes and requirements for data management change. The approach of configuring an information access system with a model allows rapid, cost-effective evolution of system capabilities. The approach of configuring an information access system with an information model allows rapid, cost-effective evolution of system capabilities.

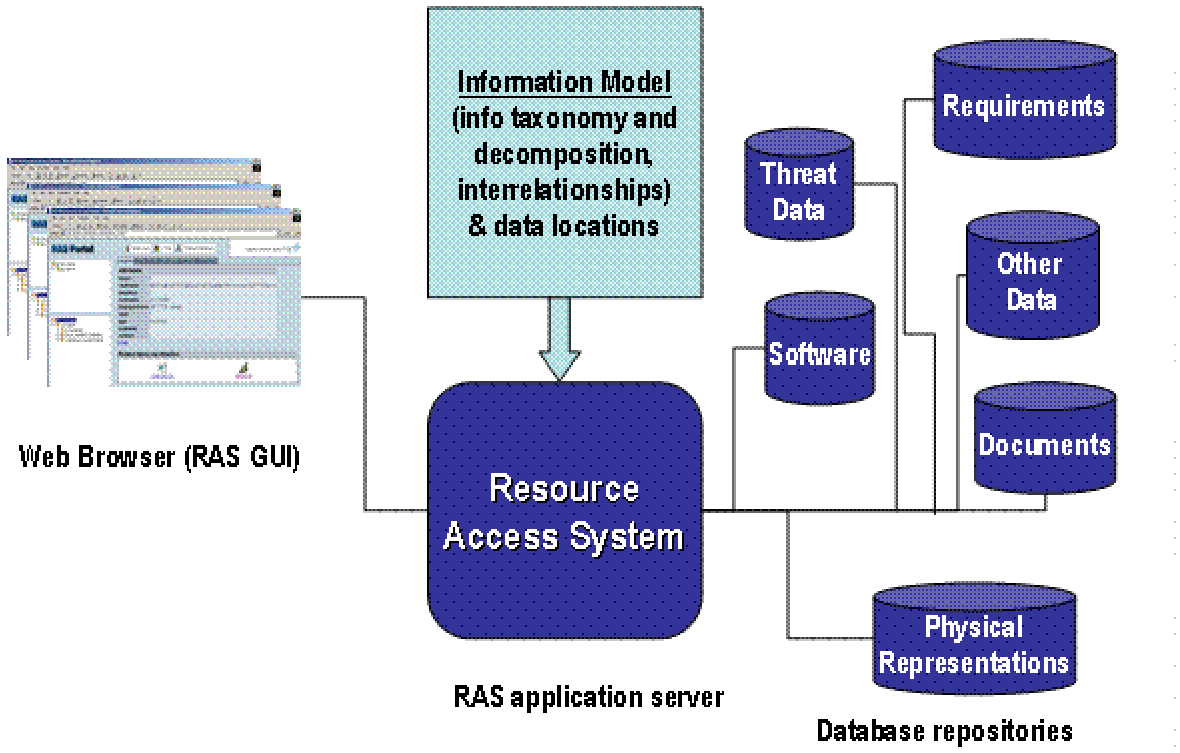


Figure 20: A Web-based System for Product Information

4 Measuring Enterprise Maturity

The premise of this paper is that the maturity of product development effort can be correlated with the capability to provide accurate, authoritative data in the form needed when required. To satisfying the information requirements they must be well understood, explicitly stated, and then implemented by the collaborative environment. Enterprise maturity can be gauged in terms of the development and analysis of the product development operations architecture. Further, constructing operational models is the best way to capture operational requirements. Constructing a realistic operational model is a difficult task as it is difficult to collect the information needed to construct a good model. The DoDAF work products are particularly useful in collecting the information needed to construct operational models. If an organization has gone to the trouble to develop such work products it is in a much better position to understand the operational requirements and provide a roadmap to satisfy the requirements.

Modeling product development operations, using the model to derive information technology requirements, and mapping the requirements on the system (physical) implementation is an indicator of enterprise maturity. The work products produced as part of this analysis process can be used to establish metrics to measure product development maturity and predict success. Our overall conclusion is that DoDAF is an extremely effective tool for product development analysis. The DoDAF products have been used to develop information models, determine who produces what data, where data comes from, and need dates. Developing the

DoDAF architecture view products allowed us to pinpoint deficiencies and gaps that otherwise would have been hard to understand.

4.1 DoDAF products can be used to measure capability maturity

As is well known, obtaining quantifiable measures that accurately predict risk of cost and schedule slippage, and technical integrity issues is difficult. The use by a program of DoDAF to analyse product development is in itself a measure of program technical maturity. Ideally the analysis described here should be performed before a program starts. Acquisition agencies could request the DoDAF products, e.g., the enterprise models, analysis and plans for infrastructure as a precondition for program award. The DoDAF products serve as evidence that the product development system has been adequately thought out and plans are in place to bring the system in compliance with requirements. The most critical questions concerning program maturity assessment can be answered by reviewing the logical data model, the activity model, and the mapping of the logical model onto the physical model.

4.2 Lessons Learned

The gaps and discrepancies that we have encountered result primarily because incompleteness of the operational model leads to incompleteness in the physical collaborative infrastructure required to realize the operational model. Many organizations do not have a collaborative information technology environment; rather they have network connectivity and a collection of unintegrated tools and processes. In fact many organizations do not explicitly recognize the need for collaborative infrastructure beyond tools and processes. Operational View use cases identify network connectivity requirements, as well as, access control and the need for mechanisms to segregate data for privacy and security. Use cases also identify needs to combine data from multiple sources. The result is that with only a manual ability to access and combine data from multiple databases an immense amount of time can be spent finding and reformatting data produced by one tool for use in another tool. The analysis of why such situations occur gives insight into how to fix the problems. Not all of the causes are related the collaborative environment deficiencies. Some of the causes result from insufficient planning to produce data, unclear roles and responsibilities, and unclear policies.

4.3 Conclusion

In the course of performing the kind of analysis described in the paper it has become clear that product development enterprise modeling can be integrated into the enterprise information management system and serve as a tool for managing technical progress and hence controlling the development process. For example, development activities can be represented with their input and output conditions with state descriptions of progress. The activity state descriptions are directly interpreted within the system by the existence of documentation of exit criteria having been met.

5 References

Department of Defense Architecture Framework Working Group, Department of Defense Architecture Framework, Volume 1, February 2004.

Graves, H., Hollenbach, J. & Barnhart M., (2003) JSF Authoritative Modelling Information System (JAMIS) Architecture (03S-SIW-040) Simulation Interoperability Workshop, September 2003.

Graves, H. & Johnson, C.., *et al* (2004) Managing Product Development Information to Ensure Data Integrity, Simulation Interoperability Workshop, September 2004.

Hollenbach, J. & Hartnett, R. (2000) The Joint Strike Fighter (JSF) Distributed Product Description (00F-SIW-077), Simulation Interoperability Workshop, September 2000.

Hollenbach, J. Bishop, S. & Graves, H. (2002), JSF Modelling Information Management, (02F-SIW-080). Simulation Interoperability Workshop, September 2002.

Scrudder, R. & Graves, H., *et al* (2003) The Critical Role of Metadata in JSF Development, (03F-SIW-097) Simulation Interoperability Workshop, April 2003.