NISTIR 5654

Defining Environment Integration Requirements

Barbara Cuthill
Marvin Zelkowitz

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Building 225, Room B266
Gaithersburg, MD 20899

NIST

NISTIR 5654

Defining Environment Integration Requirements

Barbara Cuthill
Marvin Zelkowitz

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Building 225, Room B266
Gaithersburg, MD 20899

May 1995

Table of Contents

Executive Summary

This report describes a process for defining environment integration requirements, especially data and control integration requirements, using the Information Technology Engineering and Measurement (ITEM) model for information technology use in an enterprise. This paper will discuss the use of enterprise and process modeling to classify the features of the enterprise process, its automation and the external stimuli on the enterprise that affect enterprise choices for tool and environment integration. This report will focus on the use of metadata and message types as mechanisms for integrating environments. This report includes an example of generating metadata and messaging requirements in the software development domain.

1 Introduction

Effective organizations manage the flow of information from one process to another within that organization to best use available resources. To best use available information services, organizations devote resources to automating support for information management across processes through environment and tool integration. Tool integration automates connections among specific tools. Environment integration provides the facilities needed for groups of tools to interoperate. Integrating tools and environments can automate the transition between process steps using different tools, thereby eliminating unneeded user actions, encouraging use of defined processes, and increasing productivity. This paper proposes a method for specifying environment integration requirements using the Information Technology Engineering and Measurement (ITEM) model.

The ITEM model [8] describes an enterprise's use of information technology, specifically its processes and the automation of those processes. The model defines an enterprise in terms of five "levels of abstraction." Because enterprises do not operate in a vacuum, the ITEM model also classifies external stimuli acting on the enterprise. These external stimuli encourage an enterprise to evolve by adopting new technology, lowering costs, changing business processes and quantifying success.

Defining the organization's processes and the automation to support those processes is typically part of a business process reengineering [4] effort redefining an enterprise's processes and automation to better match the enterprise's goals and take full advantage of available technology. Business process reengineering encompasses the continuing evolution of the enterprise to meet enterprise goals through improvements in processes and the automation that implements those processes.

Within a business process reengineering effort, tool and environment integration is a response to process requirements. For example, if a software development process includes developing test cases from requirements, a test case generation tool could use the products of the requirements analysis tool. The requirements analysis tool might send a message to the test case generation tool containing requirements for testing, or the requirements analysis tool might store completed requirements definitions in a repository using a standard representation accessible to the testing tool. The integration method chosen depends in large part on the policies and procedures of the organization.

The proposed technique of using the ITEM model to collect and organize requirements helps the environment integrator improve the requirements collection process thereby decreasing the cost of integration efforts. Improved requirements decrease costs over the entire development life cycle by decreasing the number of costly errors and miscommunications in later phases of the life cycle.

The rest of this paper will provide background on integration and the ITEM model (sec. 2), the proposed method for defining integration requirements (sec. 3), an example use of the proposed method (sec. 4) and conclusions (sec. 5).

2 Background

2.1 Integration Definitions

While integration is a response to process needs, there are several accepted classes of integration responding to different process needs. *Presentation integration* imposes consistent user interfaces on a set of tools reducing requirements on the user to learn and maintain different interface commands. *Data integration* allows tools to share information or work cooperatively on the same data permitting users to employ multiple tools on shared data more effectively. *Control integration* encompasses the mechanisms tools use to exchange and enact events (i.e., starting or ending of particular jobs or tasks). Each integration class provides different, complementary benefits to the user and imposes different requirements on the supporting environment. While these integration classes are clearly related, they are frequently discussed separately to provide different perspectives on the problem of integrating tools.

There are two primary approaches to implementing an integration class. *Point to point integration* connects tools through specialized conversions or private agreements on interfaces and conventions resulting in tightly integrated tools which can share and understand common data and recognize commands. Because each tool pair may require a separately maintained interface, there is a high maintenance cost associated with this model. *Framework integration* provides interfaces to common data sharing, communication, user interfaces, or other integration mechanisms. Because each tool, theoretically, requires only an interface to the framework instead of interfaces to every tool, a framework integration approach should decrease the long term integration costs. The *Reference Model for Frameworks of Software Engineering Environments* (Framework RM) [7] defines a framework as "the [relatively] fixed infrastructure capabilities which provide support for processes, objects, or user interfaces" [7].

2.2 Requirement Specification and Collection

Integration efforts for any class or approach are software development efforts and should follow the same software development practices as other software development efforts. This paper focuses on the requirements collection and analysis and the high level design portion of an integration effort. Software requirements analysis is the initial phase of the traditional software development life cycle. The product of this phase is the software requirements specification (SRS) [3] which defines attributes of the software under development. The purpose of the SRS is to communicate the needs and wishes of the eventual buyers and users of the software to the software developers and testers. Davis [3] lists several properties of a well-written SRS: consistency, completeness, nonambiguity, verifiability, understandability by non computer specialists, modifiability, traceability and annotation [3]. These goals require a well-organized set of requirements clearly related to the eventual uses of the software.

A good SRS can aid in detecting errors both during requirements analysis and later in the development cycle. Detecting and correcting mistakes, omissions, inconsistencies and ambiguities early in the life cycle is much less expensive than correcting errors in later products. The SRS also provides guidance to the software testers about what is important to the software users, and, therefore, should be tested extensively.

For the SRS to be useful in detecting errors, it must be well-organized and clearly written. Davis defines as the most difficult part of requirements analysis "organizing all the information relating different people's perspectives, surfacing and resolving conflicts, and avoiding the internal design of the software." [3] We propose that use of the ITEM model for requirements specification, organization, collection and analysis can begin to address these problems.

2.3 ITEM Model

The ITEM model [8] describes an enterprise's processes, automation for those processes and the relationship between its processes and automation at five levels of abstraction. The ITEM model also defines four sources of external stimuli on the enterprise: *market forces*, *technology changes* available to the enterprise, *models* that the enterprise uses, and *measurements* of the enterprise's actual processes and products. The ITEM model was developed from previous work on the *Reference Model for Frameworks of Software Engineering Environments* [7] and the *Project Support Environment Reference Model* [1].

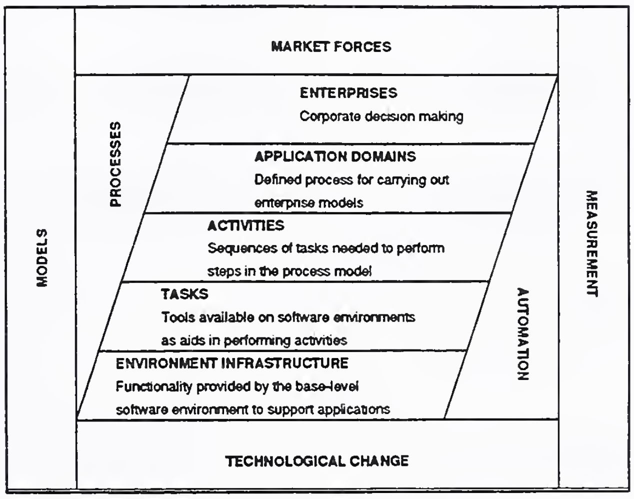
MARKET FORCES
ENTERPRISES
Corporate decision making
APPLICATION DOMAINS
Defined process for carrying out enterprise models
ACTIVITIES
Sequences of tasks needed to perform steps in the process model
TASKS
Tools available on software environments as aids in performing activities
ENVIRONMENT INFRASTRUCTURE
Functionality provided by the base-level software environment to support applications
TECHNOLOGICAL CHANGE
MODELS
PROCESSES
AUTOMATION
MEASUREMENT

Figure 1: ITEM Model

The ITEM model uses the following five information levels or levels of abstraction, for describing processes and automation within an organization [8]:

- The **enterprise level** defines organizational policy and decision making.
- The **application domain level** defines the methods for implementing the enterprise decisions and developing specific enterprise products.
- **Activities** are sequences of steps needed to implement the application domain methods.

- **Tasks** are single steps used to carry out an activity, possibly automated using a tool.

- The **environment infrastructure** is the enterprise's information technology support services.

In mapping an enterprise to the model, each model level defines a set of components at that level. ITEM model components are processes, automation, policies, goals and resource allocations. Each level defines components which structure their subcomponents at the next level while the stimulus of available technology changes the potential capabilities of the environment infrastructure (See Figure 2). These technological changes open new options to the enterprise allowing it to evolve while external stimuli encourage that evolution.

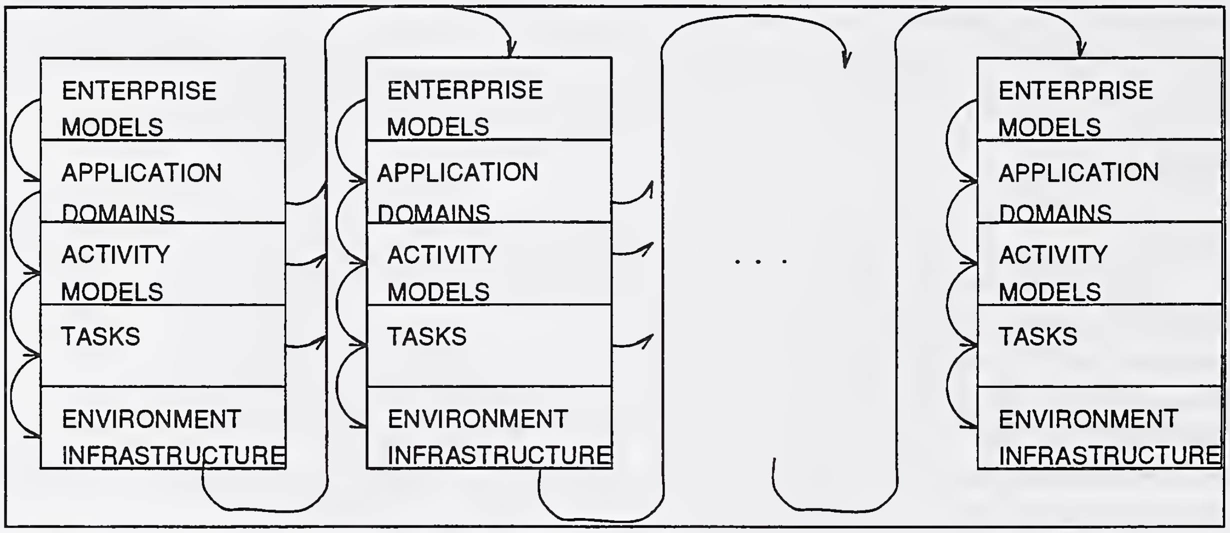

Figure 2: Enterprise Evolution and the ITEM model

Components at each level emphasize a different mix of process and automation elements. At higher levels of abstraction, process dominates the description of each component. Each lower level describes the implementation of the process and the capabilities of the enterprise in greater detail. At the lowest level, the environment infrastructure defines the basic components useful across the enterprise for automating and integrating its processes. These basic components (e.g., operating systems, email, databases) could support or automate steps in a variety of different processes. Poorly integrated tools require additional process steps or components at some level of the model to transfer information while well integrated tools can eliminate process steps and reinforce use of the process.

2.4 ITEM Model and Integration

At the activity level, the nature of the tool integration available to implement a process has the most apparent significance. An activity implements a goal of an enterprise application domain and consists of tasks which are single steps of a process. The enterprise may automate a task by using a single tool or small toolset. Tool services greatly influence task definitions. Activity level processes combine several tasks into a significant portion of an application domain process for achieving an application domain goal. Consequently, an activity may incorporate several tools into the automation of its process.

Integrating the tools supporting an activity can improve the efficiency and performance of the activity process. Integration within a task is, typically, not necessary since a task usually requires the support of only one tool or tool set. Integration at the application domain level can provide for process support across activities, but integration at this level is harder to use effectively since integration across sets of activities requires integration between much larger environments.

Components at each ITEM model level influence the enterprise's environment integration strategy imposing requirements and providing choices. Requirements generated at higher levels of the ITEM model, activity through enterprise, are process-oriented and reflect the need to automate transitions among processes and process steps. Requirements generated at the less abstract levels of the enterprise, activity through infrastructure, are automation-oriented and reflect the capabilities of the environments available to the enterprise and the difficulty of integrating specific tools. Examples of the types of integration requirements originating at each ITEM model level follow:

- The enterprise level identifies any common policies or goals for maintaining and accessing information or maintaining a common environment.
- The application domain level refines the common policies or goals of the enterprise for the development of specific products and the integration of the processes needed to develop those products.
- The activity level determines the actual tasks requiring integrated support and the availablity of integrated support alters the activity level processes.
- The automation used for implementing tasks defines the specific capabilities requiring integration.
- The environment infrastructure defines the capabilities currently available to the enterprise for integrating the tools.

Successful environment and tool integration efforts have to bridge the gap between available automation capabilities and the enterprise's defined processes. This requires incorporation of the external stimuli affecting the enterprise into the requirements. Examples of the types of requirements generated from the external stimuli follow:

- Market forces provide incentive to pursue tool integration as a mechanism for reducing cost.

- The enterprise integration model that the enterprise chooses effects its decision on any particular integration strategy.

- The technological changes available to the enterprise increase its capabilities to integrate tools.

- Measures of current process efficiency and process improvement goals effect the choice of integration strategy.

An enterprise integrates tools to provide automated support for its processes. Environment integration links enterprise processes to automated infrastructure. Process automation imposes requirements on environment supplied services. In defining an integration strategy, an enterprise has to define both its processes and its automated environment. By knowing the enterprise's processes and what capabilities the enterprise's automated environment can support, the enterprise can define any additional needs it has for environment capabilities and the connections it can use among environment components.

2.5 Enterprise Integration Options

Enterprise integration options providing extensive automated support for the process requir that tools agree on conventions for exchanging information or invoking procedures. These conventions can take the form of standardized message types and metadata (e.g., database schemas). Metadata are definitions or templates representing conventions about the syntax and semantics of the shared data. Message types provide similar definitions or templates for the messages. In an object-oriented environment, the metadata could define the components or properties of objects or classes of objects while message types could define methods for invoking objects.

Defining shared data semantics requires agreement on the underlying syntax and encodings for exchanging data. There are different data conventions depending on the complexity of the shared information's syntax and semantics. ASCII and Unicode are examples of simple text encoding schemes. Data with more complex structures require more complex encoding schemes. For example, SGML, RTF or postscript encode formatting information.

These mechanisms encode data formats; metadata conventions attempt to encode the semantics of the data. Many tools save working files in proprietary, specialized formats that allow the tool to use the information but limit outside access to it. Cooperative access to data requires that tools share a common model of the structure of the data.

Messaging mechanisms are a different approach to supporting cooperative tool usage. A tool can work cooperatively with other tools while controlling access to its data by manipulating the data in response to requests from other tools. Messaging also allows automated invocation of new tools on task completion. In some object-oriented models, messages are the methods for invoking application (or tool) objects.

Messaging and metadata approaches are overlapping approaches to the problems of supporting enterprise integration needs. Tools can use data shared through a repository or shared via messaging to create new output. These mechanisms encourage different approaches to combining information and reflect different models of developing products. Messaging and metadata approaches can be complementary in that messaging can provide for tool communication about the shared data.

The choice of integration option will depend not only on the capabilities of the organization's environment infrastructure, but on the policies of the organization as well. For example, how much effort is the organization prepared to expend in integrating particular tools? Custom integration efforts are expensive and new releases tend to require substantial maintenance on the integration code. Alternatively, buying already integrated tools may be possible, but severely limits the choice of available tools and tool features. In addition, the choice of integration mechanism for any particular pair of tools is not made in a vacuum. An enterprise may use many tools on any part of a process. For example, a software development environment may have CASE, metrics, project planning and coding tools all in use. The enterprise must decide which integrations have the highest potential payoff for the expected investment. The integration strategy for any pair of tools depends on the relative importance of the task interaction that those tools support, other tools and environments each tool must remain compatible with, the level of effort of the present mechanism and the cost of integrating those tools.

3 Defining Integration Requirements

We use the ITEM model to generate four views, termed *integration views,* of the relationships among enterprise elements and their impact on enterprise integration strategies. In combination these views help the enterprise define its integration requirements and set priorities for defining and implementing enterprise integration strategies:

* **Constraint** - This view describes restrictions on integration capabilities defined at each model level. Restrictions can derive from policy, goal, process, or automation choices.

* **Interface** - This view describes process-defined interconnections among model components.

* **Semantic** - This view describes the semantics of the interconnections among the services supporting the enterprise processes. This view begins to translate the component interconnections into requirements for automated support of those connections.

* **External** - This view describes the impact of external stimuli on the enterprise's integration strategies.

Each of the first three views provides an increasingly detailed picture of the relationships among an enterprise's constituent elements at each level of abstraction. By defining each view across all the abstraction levels, an enterprise integrator can define a comprehensive integration strategy and its impact at each model level.

The rest of this section will further define and provide an example use of these views.

3.1 Constraint view

The constraint view defines constraints that ITEM model components (e.g., goals, policies, processes, infrastructures) defined at one level impose on components at other levels. These constraints limit the range of environment integration options. Components at each level define the behavior of combinations of subcomponents at other levels and establish goals and polices restricting choices for implementing those subcomponents. The task and infrastructure levels define the enterprise's automation capabilities (e.g., infrastructure, process steps, tools) available for building supported processes. Table 1 provides typical areas for requirements on component integration defined for each level.

Task 1: Example Integration Constraints

Level	Constraints
Enterprise	Corporate goals, policies; Cost constraints
Application domains	Goal or product oriented constraints
Activities	Process constraints; organization of personnel
Tasks	Tool (both custom and "shrink wrapped" software) capabilities
Environment Integration	Infrastructure capabilities supporting integration

An earlier paper on the ITEM model [8] provided an example description of a cable broadcasting company supplying a "virtual VCR" service. This service consisted of the cable company supplying a video in response to a customer request. While the description is a simplification of the services required to supply this capability, it illustrates the ITEM model. Table 2 provides a constraint view of this example.

Table 2: Constraint view of the Virtual VCR Application

Level	Services		Constraints
Enterprise	Video on demand		Fast and accurate response to customer request; the customer is accurately billed
Application domains	Video on demand - enterprise function	Customer Relations	Customer request for service processed promptly Video library sufficient to meet customer demands
		Accounting	Customer is charged the correct amount for service
		Video Projection	Customer has simple interface to video controls Video is available at time specified
Activities	Customer Relations - application domain	Video Ordering	Customer should have simple procedure to access video library
	Video Projection - application domain	Insert Video	"Real time" response/performance required; simulation of home VCR control panel
		Play Video	
		Pause	
		Eject	
	Accounting - application domain	Pricing	Prices vary by video selected and time of day
		Billing	Update customer account; send monthly bill
Tasks	Play Video - activity	Send Images	Video projection infrastructure must support real time performance
	Pause - activity	Suspend Transmission	
	Pricing - activity	Enter customer data	Do not rekey data; receive it from customer request
		Determine price	Pricing database is accurate
	Billing - activity	Record Payment	Do not rekey data; receive it from customer request
		Bill customer	Accurately record information
Environment Infrastructure	Video projection infrastructure		Real time performance required; multiple customers processed simultaneously; process video data files
	Video Repository		Video library maintained and accessible
	Networking Infrastructure		All systems connected and able to exchange information
	MIS system		Maintain existing data; accounting data maintained

3.2 Interface View

While the constraint view describes restrictions on component integration, the interface view describes process defined component interconnections. Each process component decomposes the processes for achieving its goals into sub-processes and their interconnections for achieving sub-goals until reaching atomic task processes. These tasks generally rely on some form of automation. An enterprise could potentially automate these connections by integrating the automation supporting tasks which are related subcomponents of the same process.

The interface view focuses on two features: the information exchanged across the interface and the sequence of events required to reach that interface. In decomposing a process, components of the process may have to exchange information. This information can be products or about products of the process. A requirement for exchanging information can become a data integration requirement. Process steps, as embodied in sub-components, may execute in a particular order or require fulfillment of preconditions to execute. A requirement on the order of or conditions for invoking tools might become a control integration requirement. Table 3 illustrates interconnection requirements by providing descriptions of information exchange and event sequencing in the virtual VCR application.

3.3 Semantic View

The semantic view describes the semantics of the service interfaces by defining the metadata and messages associated with each ITEM model component. This view presumes the existence of tool and environment services supporting or supplying the associated processes. The semantic view divides into control and data integration subviews. These requirements can provide a strong link from the process requirements on the integration (e.g., link these process steps) with the implementation requirements (e.g., pass this object to this tool). Specifying the metadata and message types exchanged across each interface provides information on data and control flow for the system integrator.

This use of metadata and message types implies a standard for or a predefined agreement among vendors on metadata and message types and their binding in different concrete representations. While there are no universally accepted standards in these areas, there are several promising efforts. The Electronics Institute of America (EIA) CASE Data Interchange Format (CDIF) committee is standardizing metadata in the form of abstract description of the products of tools. ISO/IEC JTC1/SC7/WG11: Software Engineering Data Definition and Interchange project is taking these abstract descriptions and developing bindings to the Portable Common Tool Environment's (PCTE) Data Definition Language (DDL) and to the Information Resources Definition System (IRDS) Content Modules. American National Standards Institute (ANSI) Committee X3H6 CASE Tool integration methods is standardizing a set of abstract message types (termed servicegrams) for which vendors can develop bindings. These are examples of the standards committees and consortia recognizing the need for standard messages and metadata. There are many others working on standardizing metadata in specific domains or using specific formats.

Table 3: Interface View of the Virtual VCR Application

Level	Services		Information Exchanged (Data Integration)	Event Sequencing (Control Integration)
Enterprise	Video on demand		Customer relations sends video and customer information to video projection and accounting	Customer relations accepts order then notifies accounting and video projection to begin processing order
Application Domains	Video on Demand	Customer Relations	Generates order from customer	Customer begins process by placing an order
		Accounting	Requires information about requested service, prices, customer, and payment status	Updating the bill requires calculating the correct service price
		Video Projection	Video projection activities exchange transmission state and user "button pushes"	Begins with 'Insert Video' command; partial ordering to other commands
Activities	Video Projection	Insert Video Play Video Pause Halt	Exchange information on user interrupts and video transmission state among the tasks	Within each activity, first interpret user "button push" first perform legal requests
	Accounting	Pricing	Provides price given service	Done on request
		Billing	Updates bill from service info.	Update then send bill
	Customer Relations	Video Ordering	Receive information from the user and record in order form	Requires customer begin activity
Tasks	Play Video	Send Images	Provide video to customer; receive customer commands	Perform tasks in correct sequences for smooth video transmission and response
	Pause	Suspend Trans.		
	Pricing	Enter customer data	Receive data in a form that can be entered into database	Requires customer request to perform task
		Determine price	Receive information to index pricing database	Done on request
	Billing	Bill customer	Uses account status information	Done at regular intervals
		Record Payment	Billing system records bank and customer information	Respond to customer payment
Environment Infrastructure	Video projection infrastructure		Support video transmission of videos to and feedback from individual customers	Respond to customer control (VCR controls) of presentation delivery
	Video Repository		Maintain videos	Respond to requests
	Networking Infrastructure		Support enterprise policies on application data sharing	Support enterprise processes on task (application) ordering
	MIS systems		Support needed business data management	Respond to application request for information

Messaging and metadata requirements are not equivalent to control and data integration requirements. Messaging requirements place requirements on the infrastructure and tools to support the exchange of specific types of messages. Messaging is only one mechanism for implementing control integration which includes any mechanism that permits the interaction of multiple tools on a single process without necessarily sharing any common view of that process.

Metadata requirements define types of data or views of common data that the tools should possess. Defining metadata is only one mechanism for achieving data integration which includes any mechanism for sharing data across multiple tools. Data and control integration approaches are not mutually exclusive and requirements on each are often difficult to separate.

Table 4 provides a set of possible metadata and message type definitions for the Virtual VCR application example.

3.5 External View

Exchanging messages and sharing metadata may both satisfy the requirements for tool interaction to support the process. The choice of which approach to take is not entirely dependent on the original requirements. There are concerns external to the enterprise which influence the decision about which approach to pursue.

External stimuli impact all decisions of the enterprise including decisions to support a process by integrating tools used in that process. External stimuli provide incentives for increasing integration through market pressure for increased productivity and new technical innovations available to the enterprise. Alternatively, external stimuli can also provide disincentives for integration through measures of the costs of the integration and the difficulty of maintaining a complex integrated environment. Table 5 lists examples of the external stimuli which can influence enterprise integration decisions.

Table 4: Semantic View of Virtual VCR Application

Level	Services		Metadata identified	Messages identified
Enterprise	Video on demand		Customer Order	Service Request Message
Application domains	Video on demand	Video Projection	Customer Order (info used: video, time of broadcast, customer info)	Service Request Message Service Status Message
		Accounting		
		Customer Relations		
Activities	Video Projection	Insert Video	Status of broadcast Customer order (info used: video, time of broadcast, customer info)	Insert Video Request Msg
		Play Video		Play Video Request Msg
		Video Pause		Pause Video Request Msg Restart Video Request Msg
		Video Halt		Stop Video Request Msg
	Accounting service	Pricing service	Customer Order (info used: video, time of broadcast, price)	Pricing Request Msg
		Billing service	Customer Order (info used: customer, price) Customer Bill	Billing Request Msg
	Customer Relations	Video Ordering	Customer Order	Service Request Msg (initiated)
Tasks	Play Video	Image sending	Status of broadcast	Stop Trans. Request Msg Start Trans. Request Msg
	Video Pause			
	Pricing	Data Query	Customer Order (info used: service ordered, price)	Pricing Request Msg
		Calculation	Price	Calculation Request Msg
	Billing	Data Storage	Customer Account	Account Status Req. Msg Update Account Message
		Reporting	Customer Bill Customer Payment	Send bill message
Environment Infrastructure	Video projection infrastructure		Status of Broadcast	Status request message Update status message
	Video Repository		Videos	
	Networking Infrastructure		Status of IT services	
	MIS system		Databases, Accounts, etc.	

Table 5: External Stimuli and Integration

Category	Examples
Market Forces	1) Reduce costs/improve productivity 2) In home channel capacity 3) Increase library capacity
Technological Change	1) Availability of integration standards and standards compliant products 2) New application and environment features 3) Increased transmission bandwidth available 4) Use of digital data transmissions and video standards 5) Use of video compression technology 6) Transmission capacity available via cable, fiber optics, or satellite
Models	1) Process model that integration should support 2) Integration model (e.g., framework, point to point) that enterprise uses
Measurement	1) Expected savings/productivity increases from integration 2) Cost of data transmission per video 3) Simultaneous video transmissions

3.5 Requirements Definition

Each view provides a description of the relationships among enterprise components. These views classify information about the relationships, organizing it into a form that a system integrator can use for requirements analysis and high level design. The ITEM model views provide a mechanism for specifying and organizing internally and externally imposed constraints on integration strategies, requirements on component interfaces, and process imposed requirements on the semantics implemented by those interfaces. The system integrator can use this information to collect requirements for an integration strategy and trace those requirements back to functional units within the enterprise.

Davis [3] identified four areas that are difficult in requirements analysis:

* Organizing all the information
* Relating different people's perspectives
* Surfacing and resolving conflicts
* Avoiding the internal design of the software

The ITEM model provides a structure for organizing information about an enterprise's processes and automation and classifying it both by level of abstraction and functional area of interest. It handles different perspectives by allowing activities to combine overlapping sets of common tasks. Activities can reflect different sets of job functions while the underlying tasks remain constant across many activities (e.g., update database). Since tasks and infrastructure tend to be consistent or fairly consistent across the enterprise, conflicts in the use of the infrastructure or tasks will tend to surface in conflicting requirements on those tasks and infrastructure at the activity and domain levels. Finally, the model does not address internal design issues, only the external behavior that the requirements exhibit.

The model facilitates collecting requirements from different levels of abstraction which impact the problem of integrating the same components. Examination of these collected requirements for a particular set of components permits the system integrator to focus on the modeled enterprise features influencing the choice of integration strategy. For example, Table 6 provides the collected views of the tasks in the billing activity in the virtual VCR application. Task level interconnections correspond generally to tool level interconnections which makes tool interconnections supporting a frequently performed activity a good candidate for integration. Several of these requirements obviously interact. While the ITEM model helps to surface that interaction, it is still up to the system integrator to identify problem areas. Duplicated or partially duplicated requirements can indicate a very strong requirement clearly traceable through the enterprise's process and automation choices to multiple sources.

Description of the integration requirements at all ITEM model levels is important for understanding why the enterprise values this integration and what the integration effort actually entails. Without the process level information, it is difficult to understand why the enterprise values the integration or what purpose the integration serves. Without the details of what pieces of the software environment the enterprise proposes to integrate, it is difficult to judge the cost or complexity of the integration effort.

The enterprise can use all model description levels to relate external stimuli to the integration effort. These external stimuli describe both the technological advances (technological change) available to the enterprise and the pressures to adopt less costly processes (market forces) for producing products. The ITEM model external stimuli also includes the cost benefit models available to the enterprise for determining the value of carrying out the integration effort and the metrics for verifying that these models apply. It is the ability to measure and model the process, current automation and projected integration that most require understanding the ITEM model description of the integration requirements. The ability to make these measurements permits the calculation of the costs and benefits of the integration.

Table 6: Collected ITEM Views of Billing Process

Service	Constraint view	Interface View		Semantic View	
		Information Exchanged	Event Sequence	Metadata	Message Types
Enterprise - Video on Demand	Fast, accurate response to customer with accurate customer billing for service	Customer relations sends video and customer information to accounting	Customer relations has to accept the order then notify accounting	Service Request	Add New Charge
Application Domain - Accounting	Customer has to be charged the correct amount for service	Pricing activity has to receive sufficient information to price the service and send the price to billing	Billing requires previous calculation of new charges	Service Request - Customer - Video - Time	Add New Charge Calculate Price
Activity - Billing	Bill for accumulated charges sent each month (hypothetical billing policy)	Uses price and customer information to update customer charge	Updates bill with price information for regular billing to customer and recording of customer payment	Service Request as above Customer Order - Customer - Price Monthly Bill - Customer - Accumulated Charges	Add New Charge Send Bill Record Payment
Task - Billing tasks	Avoid rekeying data; provide accurate records	Billing database receives new charges, produces bills for customer, and gets updated with payment info	Each task done in response to specific customer action or at timed intervals	Service Request Customer Order Monthly Bill as above with data types for each data element	Notification messages that - Charges updated - Bill sent - Payment recorded
Environment Infrastructure - MIS systems	maintain existing data	Support database needs of billing activity	Respond to application information request or update	Support for structuring customer records as above	Support for sending and receiving the messages above

4 Example: Software Development

This section provides an extended example using the ITEM model to define requirements in the software development domain. This example focuses on the tasks and activities typically considered part of software development, and the integration requirements that those tasks and activities can impose on software engineering environments (SEE) supporting them. Table 7 provides a outline using the ITEM model of the example enterprise. It is an enterprise which performs software development, and the only portions of the enterprise that matter for this example are those directly connected to the production and maintenance of software. The example concentrates on the task and activity levels of the model to focus on the problems of defining integration requirements for SEEs while ignoring other parts of the enterprise.

Table 7: Software Development Enterprise

Level	**Description**
Enterprise	Software company
Application Domain	Software Development using Waterfall Life Cycle Model
Activities	example activities as defined in DIS 12207-1
Tasks	as defined for activities in DIS 12207-1 tool support for tasks as appropriate using definitions from the PSE RM
Environment Infrastructure	POSIX compliant heterogeneous platforms messaging infrastructure repositories

For this example, we will use the software development activity definitions from DIS 12207-1: Software Lifecycle Processes. This is a draft international standard defining the major software development processes and the tasks composing those processes. We used DIS 12207-1 because it provides a convenient set of accepted definitions for these activities. DIS 12207-1 does not require the use of a specific life cycle model to organize the processes or inclusion of all the processes. Each enterprise must define its life cycle model and assemble a tailored subset of the activities into its software development process. This example will use the 12207-1 life cycle process definitions as the generic software development activities found in an example enterprise. DIS 12207-1 defines the following "software lifecycle processes" as software development activities:

1) Process implementation
2) System requirements analysis
3) System architectural design
4) Software requirements analysis
5) Software architectural design
6) Software detailed design

7) Software coding and testing
8) Software integration
9) Software qualification testing
10) System integration
11) System qualification testing
12) Software installation
13) Software acceptance support

In the ITEM model, the application domain defines the relationships among its component activities including its life cycle model, the order of activities, the products of activities and the interaction of these activities with other activities such as configuration management. For this example, we decided to use a "waterfall life cycle model" with each activity completing before the next can begin and the products of each activity taken as input into the next activity. Each activity also interacts with a configuration management activity to manage both the final and any intermediate products of the activity. We selected this model because it is widely used and other models such as rapid prototyping generally use either abbreviated forms of the waterfall model or use multiple iterations of the model. While we selected this model, there are many other life cycle models and options for tailoring the choice of activities and assembling them into a software development process. The example software development application domain defines the interactions among the activities affecting the information exchanged and the sequencing of the activities. The system integrator can derive data and control integration requirements from these activity interactions.

Activities are sequences of tasks and ITEM model activity definitions include definitions of task products and interactions. For this example, we use DIS 12207-1 definition of the process tasks comprising the software development activities. Typically, tools support tasks, and for this example, we map the software development tasks to generic tool support defined in terms of services in the Project Support Environment Reference Model (PSERM) [1]. It is the interfaces between these services and the information and control flow across these interfaces that is the concern of the system integrator and the software engineering environment developer. The following is a list of the tasks and associated services supporting those tasks for each of the DIS 12207-1 software development activities:

System Requirements Analysis

Analyze intended use of the system - *System requirements analysis service (PSERM 4.1.1)*

Evaluate system requirements for traceability, consistency, testability and feasibility - *System requirements analysis service (PSERM 4.1.1)*

System Architectural Design

Establish top level architecture - *System design and allocation service (PSERM 4.1.2)*

Evaluate architecture for traceability, consistency, appropriateness, and feasibility - *System design and allocation service (PSERM 4.1.2)*

Software Requirements Analysis

Establish software requirements - *Software requirements analysis service (PSERM 4.2.1)*
Evaluate requirements for traceability, consistency, testability, and feasibility - *Software requirements analysis service (PSERM 4.2.1)*
Conduct review

Software Architectural design

Transform requirements into architecture - *Software design service (PSERM 4.2.2)*
Develop top-level interface design - *GUI building service*
Develop design for database - *Database design service*
Develop preliminary user manuals - *Publishing service (PSERM 7.2)*
Develop preliminary test requirements and schedule for software integration - *Software testing service (PSERM 4.2.9), Planning service (PSERM 6.1)*
Evaluate design for traceability, consistency, appropriateness and feasibility - *Design simulation and modeling service (PSERM 4.2.3)*
Conduct review

Software Detailed design

Develop a detailed design for each component - *Software design service (PSERM 4.2.2)*
Develop a detailed design for database - *Database design service*
Update user's manuals - *Publishing service (PSERM 7.2)*
Define and document test requirements and schedule for software units - *Software testing service (PSERM 4.2.9), Planning service (PSERM 6.1)*
Update test requirements and schedule for software integration - *Software testing service (PSERM 4.2.9), Planning service (PSERM 6.1)*
Evaluate detailed design for traceability, consistency, appropriateness and feasibility - *Software design service (PSERM 4.2.2), Software simulation and modelling service (PSERM 4.2.3)*
Conduct review

Software coding and testing

Develop each software unit and database - *Software generation service (PSERM 4.2.5), Compilation service (PSERM 4.2.6)*
Test each software unit and database - *Debugging service (PSERM 4.2.8), Software static analysis service (PSERM 4.2.7), Software testing service (PSERM 4.2.9)*
Update the user's manuals - *Publishing service (PSERM 7.2)*
Update the test requirements and schedule for integration testing - *Software testing service (PSERM 4.2.9), Planning service (PSERM 6.1)*
Evaluate software code and test results for traceability, consistency, test coverage, appropriateness and feasibility - *Software static analysis service (PSERM 4.2.7)*
Conduct review

Software Integration

Develop integration plan - *Planning service (PSERM 6.1)*

Integrate software units and test aggregates - *Software build service (PSERM 4.2.10), Software testing service (PSERM 4.2.9)*

Update user's manuals - *Publishing service (PSERM 7.2)*

Develop and document test cases for qualification requirements - *Software testing service (PSERM 4.2.9)*

Evaluate integration plan, design, code, tests, test results and user's manuals for traceability, consistency, test coverage, appropriateness, conformance and feasibility - *Software testing service (PSERM 4.2.9), Software static analysis service (PSERM 4.2.7)*

Conduct review

Software Qualification Testing

Conduct qualification testing - *Target monitoring service (PSERM 4.1.9), Software testing service (PSERM 4.2.9)*

Update user's manuals - *Publishing service (PSERM 7.2)*

Evaluate design, code, tests, test results and user's manuals for test coverage, conformance, and feasibility -

Support audits

Update and prepare deliverable software for integration testing - *Debugging service (PSERM 4.2.8)*

System integration

Integrate units into system - *System integration service (PSERM 4.1.6)*

Develop system qualification tests and test procedures - *System testing service (4.1.5)*

Evaluate integrated system for test coverage, appropriateness, conformance, and feasibility

System qualification tests

Conduct system qualification tests and record results - *System testing service (4.1.5)*

Evaluate system for test coverage, conformance and feasibility

Support audits - *Traceability service (PSERM 4.1.10)*

Software Installation

Develop a plan to install the software - *Planning service (PSERM 6.1)*

Install the software - *Tool installation and customization service (PSERM 7.4.1)*

Software Acceptance Support

Support acceptance review

Complete and deliver documentation - *Publishing service (PSERM 7.2)*

Provide training

This set of activities and tasks including the tool services supporting those tasks provides the context for the software engineering environment integration work in this example. The enterprise would like to improve its productivity by providing as much automated support for the software development process as possible. To reach the full potential of automating the process, however, requires also automating the connections between process steps, that is integrating the tool services supporting the process.

4.1 Constraint view of the Software Development Enterprise

The constraint view of the enterprise defines the restrictions on the enterprise imposed goals and restrictions on the integration of processes and automation. Table 8 provides the constraint view of the example software development enterprise.

Table 8: Constraint view of Software Development Enterprise

Level	Requirements
Software Development Enterprise	1) The enterprise defines limitations on cost of the integration.
Software Development Application Domain	1) The application domain uses the Waterfall Life Cycle Model. 2) The same tool service used in different activities should be supplied by the same tool (i.e., minimize the number of tools).
Software Development Activities	1) Tasks should be carried in parallel within an activity when possible.
Software Development Tasks	1) Tools should all run on common platforms.
Environment Infrastructure	1) The infrastructure is standards compliant. 2) The infrastructure includes a repository and messaging infrastructure

These requirements focus on the overall costs of the integration effort and characterizes in general terms, the connections among enterprise components.

4.2 Interface View of the Software Development Activities

DIS 12207-1 defines software activities and tasks but does not impose interconnections on those activities. The user organization tailors the activities and tasks to a specific life cycle model. For this example, we will use the waterfall lifecycle model. Interfaces exist between *process defined* activities or tasks that must exchange information or control flow. If there is tool support for activities and tasks which have a process defined relationship, then there is an interface between the tools supplying services for those activities and tasks. The enterprise may or may not automate that interface by integrating the tools depending on the relative costs and benefits of that integration. It is these interfaces that are the concern of the system integrator. Providing better interfaces between tools and supporting process automation is the purpose behind integration efforts. For this example, we will assume that the software development enterprise uses a waterfall life cycle model in which each activity follows in sequence upon completion of key the tasks in the previous activity. Table 9 classifies the information exchanged across activity interfaces and the sequence of events leading to use of an activity interface.

Table 9: Interface View of Software Development Activities

Activity	Information Exchanged	Event Sequencing
Process Implementation		
Sys. Requirements Analysis	Requires information on intended use of system Produces system requirements	Starts after process initialized
Sys. Architectural Design	Requires system requirements Produces system architectural design	Starts after system requirements established and validated
S/W Requirements Analysis	Requires system architectural design, system requirements, and information on intended use Produces s/w requirements	Starts after system architectural design established and validated
S/W Architectural Design	Requires s/w requirements Produces s/w architectural design, interface design, database deign, user manuals, and s/w integration testing requirements and schedule	Starts after s/w requirements established and validated
S/W Detailed Design	Requires s/w architectural design, interface design, database design, user manuals, s/w integration testing requirements and schedule Produces detailed design for application and database, updated user manual, s/w unit testing requirements and schedule, and updated s/w integration testing requirements and schedule	Starts after s/w architectural design, interface design, database design and user manuals are validated
S/W Coding and Testing	Requires detailed design for application and database, updated user manual, s/w unit and integration testing requirements and schedule Produces tested s/w units, updated s/w integration testing requirements and schedule and updated user's manuals	Starts after updated detailed design, user manual, and s/w integration and unit testing requirements and schedule are validated
S/W Integration	Requires tested s/w units, updated user manual, and updated s/w integration test requirements and schedule Produces s/w qualification testing requirements and schedule and tested integrated s/w units	Starts after updated user manual, s/w integration testing requirements and schedule and s/w units are validated
S/W Qualification Testing	Requires tested integrated s/w units, updated user's manuals, and s/w qualification testing requirements and schedule Produces tested s/w, updated user manuals	Starts after updated user's manuals, integrated s/w units, and qualification testing requirements and schedule are validated
Sys. Integration	Requires tested s/w, updated user manuals Produces system with integrated software	Starts after tested s/w and updated user manuals are validated
Sys. Qualification Testing	Requires system with integrated software Produces tested system with integrated software	Starts after system with integrated s/w is validated
S/W Installation	Requires tested system with integrated software Produces installed system	Starts after qualification testing of integrated s/w is validated
S/W Support	Requires installed system, and updated users manuals Produces accepted system	Starts after system is installed

Activity level processes require support from environments of multiple tools, and progress several products from one state to another. This complexity results in complex integration problems and interface requirements across environments. The task level description of the interface decomposes the problem since a much smaller number of tools and products are involved at any one interface. By decomposing the process, the task level decomposes the integration requirements into the requirements to integrate individual tasks and the tool services supporting those tasks, not the larger problem of integrating environments. Table 10 provides the interface view of the software development tasks.

4.3 Semantic View of Software Development Services

Finally, the semantic view of the model begins to define the data and message types that the various services supporting the different tasks will manipulate and exchange. Table 11 provides the semantic view of the software development example. DIS 12207-1 does not provide detailed information at this level. For the metadata, we provide common terms for the types of documents that each tool or service typically accepts as input, manipulates or produces. For the message types, we have used the ANSI X3H6 servicegram definitions. ANSI X3H6 has defined about 100 message types or servicegrams classified into nine groups. We have used these groups for the message type definitions.

Table 10: Interface View of Software Development Tasks

Activity	Task	Information Exchanged	Event Sequencing
Sys. Requirements Analysis	Analyze sys. use (T1)	Requires information on sys. use; produces draft sys. requirements	
	Evaluate requirements (T2)	Requires draft sys. requirements; produces sys. requirements	Follows T1
Sys. Architectural Design	Create top level design (T3)	Requires sys. requirements; produces draft system architecture	Follows T2
	Evaluate against requirements (T4)	Requires sys. requirements & draft sys. architecture; produces sys. architecture	Follows T3
S/W Requirements Analysis	Establish s/w requirements (T5)	Requires sys. requirements, & sys. architecture; produces draft s/w requirements	Follows T4
	Evaluate requirements (T6)	Requires draft s/w requirements, sys. requirements, & sys. architecture; produces s/w requirements	Follows T5
	Conduct review (T7)	Requires all work to date; produces review	Follows T6
S/W Architectural Design	Transform requirements into architecture (T8)	Requires s/w requirements; Produces draft s/w architecture	Follows T7
	Develop top-level interface design (T9)	Requires s/w requirements; produces draft interface design	Follows T7
	Develop database design (T10)	Requires s/w requirements; produces draft database design	Follows T7
	Develop preliminary user manual (T11)	Requires s/w requirements & draft s/w architecture, interface design, & database designs; produces preliminary user manual	Follows T8, T9, & T10
	Develop preliminary test requirements & schedule (T12)	Requires s/w requirements, & draft s/w architecture, interface & database designs; produces preliminary test requirements & schedule	Follows T8, T9, & T10
	Evaluate design (T13)	Requires s/w requirements & draft s/w architecture, interface & database designs; produces s/w architecture, interface & database designs	Follows T8, T9, & T10
	Conduct review (T14)	Requires all work to date; produces review	Follows T11, T12 & T13

Table 10: Interface View of Software Development Tasks

Activity	Task	Information Exchanged	Event Sequencing
S/W Detailed design	Develop detailed design for each component (T15)	Requires s/w requirements & s/w design; produce draft component designs	Follows T9
	Develop detailed database design (T16)	Requires s/w requirements & database design; produces draft detailed database design	Follows T10
	Update User manuals (T17)	Requires s/w requirements, draft component & database designs, & user manual; produces updated user manual	Follows T11, T15 & T16
	Define s/w unit test requirements & schedule (T18)	Requires s/w requirements, draft component & database & preliminary s/w integration test requirements & schedule; produce s/w unit test requirements & schedule	Follows T12, T15 & T16
	Update s/w integration test requirements & schedule (T19)	Requires s/w draft component designs, detailed database design, preliminary s/w integration test requirements & schedule, & s/w unit test requirements & schedule; produces updated s/w integration test requirements & schedule	Follows T18
	Evaluate detailed design (T20)	Requires s/w requirements & draft component & detailed database designs; produces component & database designs	Follows T13, T15 & T16
	Conduct Review (T21)	Requires all work to date; produces review	Follows T17, T19, & T20
S/W Coding & Testing	Develop each s/w unit & database (T22)	Requires s/w component & database detailed designs; Produces s/w components & database	Follows T15, T18, & T21
	Test each s/w unit & database (T23)	Requires s/w components, database & test requirements; Produces tested s/w components & database	Follows T16, T18, & T22
	Update user's manuals (T24)	Requires s/w components, database & user manuals; produces updated user manuals	Follows T17 & T23
	Update test requirements & schedule (T25)	Requires s/w components, database, & s/w integration test requirements & schedule; produces updated s/w integration test requirements & schedule	Follows T23
	Evaluate software code & test schedule (T26)	Requires s/w components, database, & s/w test requirements & schedule; Produces evaluated s/w components, database & s/w integration test requirements & schedule	Follows T25
	Conduct review (T27)	Requires all work to date; produces review	Follows T24 & T26

Table 10: Interface View of Software Development Tasks

Activity	Task	Information Exchanged	Event Sequencing
S/W Integration	Develop s/w integration plan (T28)	Requires s/w integration test requirements & schedule; produces s/w integration plan	Follows T25
	Integrate s/w units & test aggregates (T29)	Requires s/w components, database & s/w integration plan; produces integrated s/w	Follows T23 & T25
	Update user manual (T30)	Requires user manual & s/w; produces updated user manual	Follows T29
	Develop & document test cases (T31)	Requires s/w testing requirements and schedule, & s/w requirements; produces test cases	Follows T29
	Evaluate products so far (T32)	Requires all products so far; produces evaluated products	Follows T30 & T31
	Conduct review (T33)	Requires all work to date; produces review	Follows T32
S/W Qualification Testing	Conduct qualification testing (T34)	Requires s/w, user manuals and test cases; produces test results	Follows T33
	Update user manuals (T35)	Requires s/w, test results & draft manuals; produces user manuals	Follows T30
	Evaluate products so far (T36)	Requires all products so far; produces evaluated products	Follows T35
	Support audits (T37)	Requires all work to date; produces audit	Follows T29
	Update & prepare software (T38)	Requires s/w and test results; produces updated s/w	Follows T36
Sys. Integration	Integrate units into system (T39)	Requires s/w & integration plan; produces integrated sys.	Follows T34
	Develop sys. qualification tests (T40)	Requires sys. requirements and sys.; produces sys. qualification test plan	Follows T39
	Evaluate system (T41)	Requires sys. & qualification test plan; produces tested sys.	Follows T40
	Support audits (T42)	Requires all work to date; produces audit	Follows T38
S/W Installation Plan	Develop a plan to install software (T43)	Requires info. on use & documentation; produces s/w instal. plan	Follows T41
	Install the software (T44)	Requires s/w instal. plan and s/w; produces installed s/w	Follows T42
S/W Acceptance Support	Support acceptance review (T45)	Requires all work to date; produces acceptance review	Follows T44
	Complete & deliver documentation (T46)	Requires all work to date; produces final documentation	Follows T44
	Provide training (T47)	Requires all work to date; produces training	Follows T21

Table 11: Semantic View of the Software Development Example

Task/Service	Tasks	Metadata	Message Types
System Requirements Analysis Service (PSERM 4.1.1)	T1, T2	System Requirement	Software Analysis and Design Servicegrams
System Design and Allocation Service (PSERM 4.1.2)	T3, T4	System Requirement System Design	Software Analysis and Design Servicegrams
S/W Requirements Service (PSERM 4.2.1)	T5, T6	Software Requirement System Requirement System Design	Software Analysis and Design Servicegrams
S/W Design Service (PSERM 4.2.2)	T8, T15	Software Requirement Software Design	Software Analysis and Design Servicegrams
GUI Building Service	T9	Software Requirement Software GUI Design	Software Analysis and Design Servicegrams
Database Design Service	T10, T17	Software Requirement Software Database Design	Software Analysis and Design Servicegrams
Publishing Service (PSERM 7.20	T11, T17, T24, T30, T35, T46	User Manual Documentation	Edit Servicegrams
S/W Testing Service (PSERM 4.2.9)	T12, T19, T25, T31, T34,	Software Requirement Software Design Software Test	Debug Servicegrams
Planning Service (PSERM 6.1)	T12, T19, T25, T28, T43	Schedule Resources	Software Process Servicegrams
S/W Design Simulation and Modeling Service (PSERM 4.2.3)	T13, T20	Software Design Software Environment	Software Analysis and Design Servicegrams
Code Editing Service	T22, T29	Software Modules	Edit Servicegrams
Software Generation Service (PSERM 4.2.5)	T22, T29	Software Modules	Debug Servicegrams
Compilation Service (PSERM 4.2.6)	T22, T29	Software Modules Executable Objects	Build Servicegrams Debug Servicegrams
Debugging Service (PSERM 4.2.8)	T22, T29	Software Modules Executable Objects	Debug Servicegrams
Software Build Service (PSERM 4.2.10)	T43	Software Modules Executable Objects	Build Servicegrams
Software Static Analysis Service (PSERM 4.2.7)	T26	Software Modules	Static Analysis Servicegrams
Tool Installation and Registration Service	T44	Executable Program	
System Integration Service	T39	Software Modules Executable	Build Servicegrams
System Testing Service	T41	Software Modules	

5 Conclusions

The ITEM model provides a useful tool for collecting and organizing software requirements especially requirements on integration capabilities. The ITEM model relates automation and process features of an enterprise to each other. By establishing the relationship of the process and automation features, it is possible to link requirements on the automation to the processes that generated those requirements.

Davis identifies three underlying principles of structuring information during requirements specification. These are partitioning, abstraction, and projection of information [3]. The ITEM model partitions information according to process defined clusters at different levels of granularity. The multiple levels of abstraction provided in the model provide a mechanism for decomposing or abstracting requirements as appropriate. The use of multiple overlapping activities and application domains to organize individual tasks permits the requirements analyzer to include different viewpoints or projections of the information.

The ITEM model has provided a promising start as a tool for requirements specification, collection and organization especially for problems in the domain of tool and environment integration. The examples in this paper, the cable company providing a virtual VCR service and a software development organization, illustrate how the ITEM model can be a tool for organizing user requirements; however, this work has only just started. The ITEM model needs to be tested further on larger projects and examples and in combination with other techniques such as formal languages and methods. The ITEM model only addresses the problem of organizing the requirements; the tool must be incorporated into more comprehensive requirements analysis techniques and software development techniques to be useful.

References

[1] Brown, A., D. Carney, P. Oberndorf, and M. Zelkowitz. *Next Generation Computer Resources: Reference Model for Project Support Environments*. National Institute of Standards and Technology, Special Publication 500-213, 1993.

[2] *CASE Data Interchange Format - Overview*. Electronics Industry Association, Interim Standard #106, 1994.

[3] Davis, Alan M. *Software Requirements: Analysis and Specification*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[4] Hammer, M. and J. Champy. *Reegineering the Corporation*. New York: Harper Collins, 1993.

[5] *Information Technology - Software - Part 1: Software Life-Cycle Process*. International Standards Organization/International Electrotechnical Commission Standard 12207-1-1994.

[6] *Portable Common Tool Environment*. International Standards Organization/International Electrotechnical Commission Standard 13719-1994.

[7] *Reference Model for Frameworks of Software Engineering Environments*. National Institute of Standards and Technology, Special Publication 500-211, 1993.

[8] Zelkowitz, Marvin and Barbara Cuthill. *Information Technology Engineering and Measurement Model: Adding Lane Markings to the Information Superhighway*. National Institute of Standards and Technology, NISTIR 5522, 1994.