IAC-08-D3.4-E5.4.5

MANAGING LARGE SCALE PROJECT ANALYSIS TEAMS THROUGH A WEB ACCESSIBLE DATABASE

Daniel A. O'Neil
Aerospace Engineer, NASA
Huntsville, Alabama, USA
daniel.a.oneil@nasa.gov

ABSTRACT

Large scale space programs analyze thousands of requirements while mitigating safety, performance, schedule, and cost risks. These efforts involve a variety of roles with interdependent use cases and goals. For example, study managers and facilitators identify ground-rules and assumptions for a collection of studies required for a program or project milestone. Task leaders derive product requirements from the ground rules and assumptions and describe activities to produce needed analytical products. Disciplined specialists produce the specified products and load results into a file management system. Organizational and project managers provide the personnel and funds to conduct the tasks. Each role has responsibilities to establish information linkages and provide status reports to management. Projects conduct design and analysis cycles to refine designs to meet the requirements and implement risk mitigation plans. At the program level, integrated design and analysis cycles studies are conducted to eliminate every "to-be-determined" and develop plans to mitigate every risk. At the agency level, strategic studies analyze different approaches to exploration architectures and campaigns. This paper describes a web-accessible database developed by NASA to coordinate and manage tasks at three organizational levels. Other topics in this paper cover integration technologies and techniques for process modeling and enterprise architectures.

ACRONYM LIST

ACL – Access Control List
API – Application Programming Interface
ATA – Architecture Trades and Analysis
AT&T – American Telephone & Telegraph
BPEL – Business Process Execution Language
BPMN - Business Process Modeling Notation
CAIT – Constellation Analysis Integration Tool
CIM – Common Information Model
CMMI – Capability Maturity Model Integration
COI – Community of Interest
ESMD – Exploration Systems Mission Directorate
DEA – Data Exchange Agreement
DFD – Data Flow Diagram
DIO – Directorate Integration Office
DMTF – Distributed Management Task Force
DRM – Data Reference Model
FEA – Federal Enterprise Architecture
ICD – Interface Control Document
IDAC – Integrated Design and Analysis Cycle
IDE – Integrated Development Environment
IPT – Integrated Product Team
IDAC – Integrated Design and Analysis Cycle
IDE – Integrated Development Environment
LDAP – Lightweight Directory Access Protocol
MDA – Model Driven Architecture
MOA – Memorandum of Understanding
MVC – Model View Controller
NASA – National Aeronautics & Space Admin.
OASIS – Org. for Adv. of Structured Info. Stds.
OMG – Object Management Group
OWI – Organizational Work Instruction
PBS – Product Breakdown Structure
RAD – Rapid Application Development
SEI – Software Engineering Institute
SOA – Service Oriented Architecture
SSO – Single Sign On
SVG – Scalable Vector Language
TDS – Task Description Sheet
UML – Unified Modeling Language
URL – Uniform Resource Locator
WBS – Work Breakdown Structure
XML – eXtended Markup Language

ACKNOWLEDGEMENTS

The author thanks the following colleagues for their contributions to the systems and activities described in this paper's case studies: Don Monell, Joshua Arceneaux, Joel Abraham, Karen Murphy, Dennis Bulgatz, Stephen Young, Jeff Crowe, Henri van den Bulk, Tom Gormley, Leland Dutro, Lisa Murphy, Adrianne Day, Sandeep Shetye, and Anita Prasad.

INTRODUCTION

Managing the integration of enterprise architecture requires orchestration of business processes, standardizing common information models, publishing reusable code and service interfaces, and establishing repeatable development and integration practices. Often, people think in terms of their discipline and express needs in that vernacular. To understand processes of several organizations, information model developers translate narrative descriptions into standardized diagrams so enterprise architects can discover common capability needs. Examples of common needs include:

- ***Project Management*** – Coordinating budgets, personnel, schedule, and products to meet requirements and mitigate risks
- ***Product Management*** – Transferring, storing, translating, and configuring within the context of a product breakdown structure that spans all levels of a program
- ***Process Management*** – Defining workflows, enabling approvals and concurrence, changing a products status, and notifying participants via e-mail
- ***Archiving*** – Aggregating designs, software, and data files into configurations and or compressing collections into files with indices and product descriptions
- ***Collaboration*** – Capturing comments, markups, revisions, discussions, and teleconference support
- ***Searching and Filtering*** – Finding information via key words, synonyms, and context within indices, product structures, and semantic information models
- ***Report Generation*** – Producing tables, spreadsheets, schedules, and diagrams that present data and relationships among people, products, and processes
- ***Information synthesis*** – Integrating data from multiple sources, plotting data, creating visualizations, and displaying results via portals and dashboards

This paper presents case studies about database applications that provide these capabilities. Other concepts covered in this paper include business process modeling, web services, model driven and service oriented architectures, graphical notations, application development tools and process. Conclusions weave these concepts into an enterprise architecture development process.

CASE STUDY 1: CONSTELLATION ANALYSIS INTEGRATION TOOL DATABASE

The Architecture Trades and Analysis (ATA) Office within NASA's Constellation Program manages Integrated Design and Analysis Cycles (IDAC). Each project (e.g., Ares, Orion) conducts Design and Analysis Cycles (DAC) composed of technical trade studies. An IDAC conducts studies to integrate results from the DACs and conducts technical trade studies at the exploration architecture level. To manage the myriad of studies, the ATA created a Task Description Sheet (TDS) to plan the study, define products, schedule reviews, identify needed skills, and establish workflows for approving and concurring on the tasks. Before December of 2006, the ATA used Microsoft Word® to create and revise a TDS. Board discussions regarding a task involved printing out hard copies of the document and projecting it on the screen. Board members would mark-up the hard copies and provide them to the task manager for integration into the TDS. The process of incorporating the comments and getting approval from the board could take days or weeks.

When hundreds of TDS were produced and scattered across several directories on the Windchill® product life cycle management system, ATA management identified the need for a web accessible database. Starting in September of 2006, the Constellation Analysis

Integration Tool (CAIT) development team rapidly prototyped a database and deployed an operational system by the end of the first week in December 2006. Now, the board reviews an individual TDS by projecting the database record on a screen, provides comments, and the task manager updates the TDS during the meeting. In 2007, subsequent deployments incorporated reporting requirements from the Directorate Integration Office (DIO) within NASA's Exploration Systems Mission Directorate (ESMD) and workflow management requirements from Level III engineering teams that support the Ares project.

The CAIT database provides a capability to create records of organizations, teams, study collections, Task Description Sheets (TDS), points-of-contact, data, and disciplines. Information on TDS includes completion status, covered requirements, identified risks and issues, models and simulations used, data providing organizations, data needs of organizations, associated ground rules and technical baselines. Establishing linkages among tasks, requirements, and risks is a primary purpose of the database. Other databases serve as the authoritative source for requirements and risk data. Computer code within CAIT imports this data to enable linking and report generation. Managers use the reports to monitor and manage design and analysis cycles within the projects or program.

THE CAIT USER COMMUNITY

Members of the CAIT user community include:

- Study Managers
- Process Facilitators
- Task Description Sheet Authors
- Discipline Specialists
- Organizational & Project Managers

INTERDEPENDENT USE CASES

The CAIT community of use cases is interdependent; each actor performs actions that depend on an action performed by another actor. Figure 1 presents the roles and responsibilities of the CAIT user community participants as well as the interdependencies of their use cases. On the left side of the diagram, a column identifies the roles and disciplines. On the right side of the diagram, a column identifies the goals pursued by the roles or disciplines. Circles within the diagram identify a particular use case and arrows between the circles depict dependencies.

Multiple scenarios derive from the relationships among the use cases. Study Managers establish study collections, define ground rules and assumptions, and assign the teams and organizations to the study collection. These use cases depend on the Organizational Manager to build teams and assign responsibilities. Goals of a Study Manager are to generate task status reports, traceability reports, and coverage reports, which depend on a Task Leader to link risks and requirements; this action in turn depends on the Study Manager to include the requirements and risks in the study collection. Traceability reports enable the study manager to explain how certain data products address a "To Be Determined" within a requirement or mitigate a particular risk. Coverage reports provide a study manager with the capability to identify all of the risks and requirements that have been addressed by studies within a collection.

A Process Facilitator uses the assigned responsibilities to create approval and concurrence paths. Facilitators identify the life-cycle milestones for a TDS and identify the specific roles that approve or concur on the activities and products described in the task and produce a workflow. Example milestones in the life of a TDS include draft, baseline, product

review, and archive. A workflow management function within the application automatically notifies the next person in the work-flow about a required review. Also, the system presents a list of TDS records that require a review, approval, or concurrence when the person logs into the system. When a TDS author describes a task, he or she uses the approval path templates. Reports generated by Process Facilitators include model and simulation traceability reports that identify the relationships between tools, data products, tasks, risks, and requirements. Other files generated by facilitators are schedules that identify activities, products, and delivery dates.

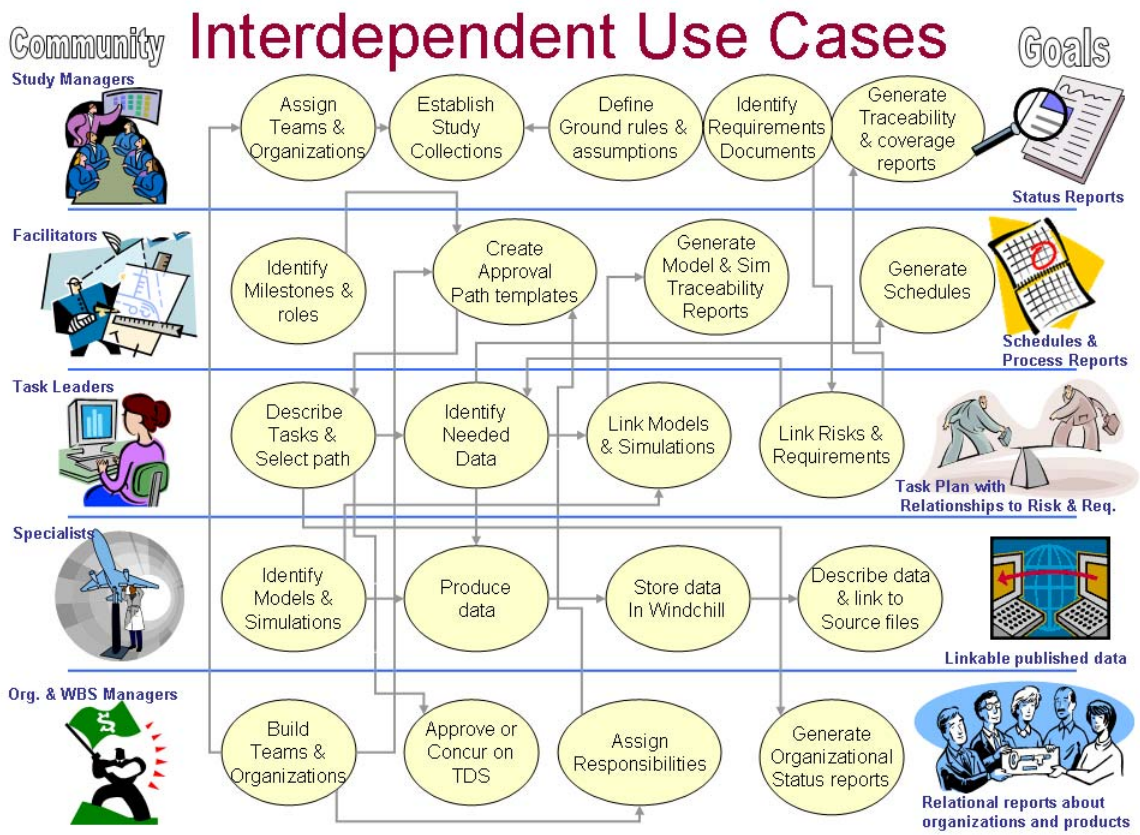


Figure 1 Interdependent Use Cases for CAIT

Task Leaders develop the contents of a TDS, identify necessary data, link requirements and risks, and identify needed discipline specialists. A TDS serves as a contract between a requesting organization and performing organization; approval and concurrence paths define the signature blocks for this contract. A report function can generate Microsoft Project® schedules generated from the product descriptions, product delivery dates, board reviews, and workflows.

Discipline Specialists perform tasks and produce products specified activities specified in the TDS. They create data-records that describe the contents and location of the product files. Often, the TDS Author will identify a need for existing data. In these cases, the discipline specialist can locate the existing files and revise the needed data record to reflect the location of those files. When an author identifies needed data that does not exist, a new data record provides a description for the

discipline specialist. Also, Discipline Specialists identify models, simulations, and analysis tools used to generate the product data. These use cases enable the Task Leaders to achieve the goal of linking tasks, products, and tools that mitigate risks or refine requirements.

Organizational Managers depend on the Task Leaders to select the approval and concurrence paths so they can participate in advancing the TDS through the life cycle. To achieve the goal of generating progress reports, the Organizational Manager and Task Manager depend on the Study Managers, Process Facilitators, and Disciplined Specialists to accomplish their activities.

CASE STUDY 2: DATA MAPPING ACTIVITY

The Constellation program established an Information Systems Office in the summer of 2008; this office manages an Integrated Product Team (IPT) composed of stakeholders throughout the program and sponsored projects. To gain insight to the existing capabilities, current processes, data flows, applications, and infrastructure, the IPT will produce "As-Is" diagrams. Initially, the diagrams will be produced by a variety of information modeling tools. Plans for this effort include the development of a centralized web-accessible database to define data exchanges and automated the diagramming process.

With a database that provides the capability to identify all the data flows across agency internal organizations, the IPT and project offices can decide, which data flows ought to be formalized. Often, organizations formalize agreements in a variety of documents ranging from a Memorandum of Agreement (MoA) to an Interface Control Document (ICD). Reports from the data exchange database can produce an agreement that can be signed by two or more parties. Generically referred to as a Data Exchange Agreement (DEA), the generated document can be tailored to define a variety of customer and supplier interfaces. Example applications of a DEA include synthesizing information from multiple sources, archiving file collections, exchanging files, or an interface between two applications.

Data Fusion or Information Synthesis

A portal or dashboard fuses data or synthesizes information from multiple sources. A team developing a portal or dashboard can write DEAs with each owner of the source data. Without having to address specific data fields and types, a DEA provides a structured mechanism for identifying needed data that can generate data flow diagrams.

Archives

Organizations exchange archives containing a variety of files. In this case the DEA provides a table-of-contents for the archive as well as descriptions of each file within the archive.

Individual files

How often do you find your e-mail stuffed with a ten megabyte status report or presentation? Typical Organizational Work Instructions (OWI) do not describe how to store or transfer data. In this scenario, a DEA can explain that instead of using e-mail to transfer data, large working files go into a Wiki and the e-mail message contains a hyperlink to the file. Organizations exchange specific types of individual files; a DEA documents the applications, file types, submission frequency and other information that enables automated diagramming of data flows.

Application Interfaces

Software applications exchange data to integrate with user inputs or calculated

values. A DEA can serve as a predecessor to a more detailed ICD. Typical ICDs have two tables that represent the two sides of the interface columns within the tables identify a variable name, type, upper and lower bounds, and update frequency.

DEVELOPING & INTEGRATING APPLICATIONS

Integrating or developing applications involves capturing requirements derived from processes, and designing, developing, and deploying the integrated code or new applications. Process models, development tools, and reusable code facilitate this process.

Process Modeling

A business process models captures organizational activities, data structures, and data flows; analyzing an "As Is" or current architecture model provides insights such as bottle necks, redundancies, inefficiencies, and other opportunities for improvement. To develop "Go To" or target architecture, determine which enterprise processes are strategic. Conduct a Kaizen event to streamline the processes; this event ought to proceed as much as possible in parallel with any enterprise architecture implementation. In addition to optimized processes, a Kaizen event can define management metrics.[1]

The Object Management Group (OMG), a non-profit industry standards consortium, originated the Unified Modeling Language (UML) as a graphical notation for object oriented software. Business Process Modeling Notation (BPMN) is another graphical standard managed by the OMG. With these notations, organizations can diagram their business processes and supporting software.[2]

The Organization for the Advancement of Structured Information Standards (OASIS) manages the standard Web Services Business Process Execution Language (WS-BPEL). This orchestration language enables organizations to specify business process behavior based on Web Services. Techniques exist for translating the BPMN diagrams into BPEL code. Since the release of the BPEL draft in the spring of 2003 all of the large vendors involved in the development have announced product support. Microsoft, IBM and BEA are all going to support BPEL based orchestration. As an orchestration language, the BPEL provides constructs to describe arbitrarily complex business processes. At the highest level, a BPEL process defines the interaction between partners. A process can interact synchronously or asynchronously with its partners, i.e., its clients and with the services the process orchestrates.[3]

The building blocks for a BPEL process are the descriptions of the parties participating in the process, the data that flows through the process and the activities performed during the execution of the process. BPEL processes can be executed via their own Web service interface, or through internal triggers defined inside the process. An external trigger is a message received on a port exposed by the process, internal triggers are time driven and defined inside the process.

Methods and Tools

Producing an "As-Is" business process model involves domain experts explaining their processes and modeling experts creating diagrams of those processes. An iterative interview, diagram, review, and revise process involves facilitators or model developers conducting interviews with groups subject matter experts. If facilitators conduct the interviews, they collect the information and provide it to the model developers. Interpreting the interview transcripts and modeling tools, the developers create the diagrams, and present them to the domain experts. To reduce time and interpretation errors, the diagramming

experts can participate in the interviews and create the models during the interview. Benefits of the process and information modeling tools include standards, validation of data and exchanges among entities, and code generation. Disadvantages include the cost and training required for sophisticated tools and the increased workload for the diagramming team as the organization researches more business processes.

The Data Mapping Activity, presented in this paper, offers an alternative approach of using a web-accessible database with form based surveys. Facilitators provide training on how to use the database and the domain experts create records to describe their processes and products. Code in the database creates graphic command scripts to produce business process and Data Flow Diagrams (DFD). Automated graph drawing tools like GraphViz, developed by AT&T Research, generate diagrams of graphs and networks from text descriptions.[4] Diagrams can be generated in the Scalable Vector Graphics (SVG) for, which is based on eXtended Markup Language (XML). Several process and information modeling tools import and export XML based files, which allow round-trip engineering between the two methods.

SERVICE ORIENTED ARCHITECTURE AND MODEL DRIVEN ARCHITECTURE

The OMG defined Model Driven Architecture standards and terminology for integrating and evolving enterprise-scale software systems; as an approach to application design and implementation, MDA encourages efficient use of system models in software development and it supports reuse within system families. Models abstract physical systems and allow engineers to focus on important details. With system models, engineers can predict system qualities, analyze the impact of changes to properties, and communicate key characteristics to stake-holders.[5]

Service Orient Architecture (SOA) allows loose coupling among software applications. A computer program, categorized as a service, performs work for another computer program referred to as a service consumer. Two architectural constraints enable services to achieve loose coupling. First, each participating software program incorporates the services' interfaces. Second, an extensible schema defines the vocabulary and structure of messages passed through service interfaces. Messages must describe rather than instruct because the service provider is responsible for solving the problem. Messages have to adhere to a format, structure, and vocabulary understood by all participating programs. Extensibility allows addition of new services and messages. A centralized registry of services enables service consumers to find service providers.[6]

APPLICATION INTEGRATION WEB SERVICES

Analysis of process models, data exchange agreements, and requirements associated with application integration requests reveal common web service needs. The CAIT development team, working with experts in enterprise architectures and service buses, identified needs for the following web services:

Common Account Request Service

Typically, each web application requires an individual to submit an account request form. If the integration framework offers several applications then filling out the request form for each application can become tedious. An ideal service is to provide a one-stop-shop for filling out a single account request form. Access privileges will have to be determined by contract and organization. Enterprise applications that span multiple computer

programs will have to be aware of the user's privileges when accessing various databases.

Access via the Central User Interface

Applications presented via centralized Graphical User Interface (GUI) or web portal as a part of the integration framework must have a process for providing help-desk support. If an application is linked to the integration framework user interface and someone has a problem, they will probably call the help desk for the integration framework. The help desk needs instructions or a script to assist the user with basic problems such as resetting a password. For more difficult problems, the help desk needs contact information for transferring the call to the development or operations team for application.

Account Authentication Service

A central authoritative source of information about civil service and contract personnel would prove to be a powerful integration tool. An external account authentication service would enable other application developers to check accounts against a centralized Lightweight Directory Access Protocol (LDAP) database contained in or accessed by the integration framework. Using the account authentication service, external applications would send a request to the integration framework to check the profile of a user. A response from the integration framework would indicate whether the incoming credentials match the profile in the framework's LDAP database. If a corresponding record does not exist in the authoritative authentication database then a verbose error message is returned to the external application.

Single-Sign-On Service

Doesn't get frustrating to have to enter a different user name and password each time you open a different application? A Single-Sign-On (SSO) service, for tightly coupled applications, would use the integration framework's LDAP to authenticate user access. For external loosely integrated applications, the integration framework could provide an SSO services that enable users to synchronize their user names and passwords with the framework's authoritative LDAP database. Working the account authentication service, this service receives a request from the external application. The external application's Access Control Lists (ACL) has accepted the log-in and the framework has checked the username and password against the LDAP. This SSO service automatically logs into the framework so the user does not have to enter the credentials again. In the near term, this service could be one-way, meaning that the SSO only works if the person logs into the framework GUI or portal first. Eventually, the service could be bi-directional, meaning that a person could log-into the framework GUI or the external application.

External Application Data Transport Service

An enterprise service bus within the integration framework enables applications to exchange data via a common interface. This capability reduces the N^2 number of interfaces to N number of interfaces because each application development team only has to create an interface for the Enterprise Service Bus (ESB). Extending the data exchange services to external applications involve publishing an Application Programming Interface (API) for the integration framework. The API should provide sample code and bindings to a variety of languages.

External Application File Upload Service

If the integration framework contains a tightly coupled document repository, it can provide a service to exchange files with

other applications in the framework. For external, loosely applications, the integration framework can provide a file exchange service that works with the account authentication service to verify access rights of the user. Referencing a PBS, the file exchange service could determine where to upload the file in the repository and return a Uniform Resource Locator (URL). Applications could have a drop-box where users could bulk-upload or drag-n-drop files. A polling loop within the application could periodically call the file exchange service. The service could determine where to store files based on a naming convention. If a file does not meet the naming convention, the service returns an error. To retrieve files from the repository hosted libraries, the application would request a file via the URL, the file exchange service verifies access rights with authorization service and allows the repository to download the file.

CAPABILITY MATURITY MODEL

Watts Humphrey developed the Capability Maturity Model (CMM) in 1987 for the Department of Defense's Software Engineering Institute (SEI) at Carnegie Mellon University.[7] The most recent version, CMM Integration (CMMI), defines five levels of maturity for software development processes.

Level One: Processes have ad hoc approaches, methods, notations, tools and produce unpredictable results. Management tends to be reactive and success is highly dependent on the skills of the team.

Level Two: processes are repeatable. The organization applies discipline to managing requirements, planning projects, monitoring and controlling processes. Processes include managing supplier agreements and configurations and assuring product and process quality through measurement and analysis. Processes focus on project-level activities and practices.

Level Three: Processes are defined and documented with consistent, cross-project disciplines to establish organization-level activities and practices. Efforts emphasize evolution of requirements from multiple stakeholders, evolutionary design, continuous integration, and change management. Management plans include verification, validation, risk management, training, and decision analysis and process definition.

Level Four: processes are quantitatively managed. Historical results for Level Three projects can be exploited to make trade off, with predictable results among competing dimensions of business performance (cost, quality, timeliness). Focus areas include performance setting, benchmarks, and project management based on statistical quality control methods.

Level Five: Processes are optimized and rapidly reconfigurable. The organization learns, adapts, and continuously improves through quantitative assessments. Focus areas include causal analysis and resolution, proactive fault avoidance and best practice reinforcement.[8]

Reusable Code Libraries

Given the recurrence of required functions such as security, search, agreements, synthesis, workflows, and report generation, planning and designing code for reusability can save development costs in future projects. Ongoing integration activities associated with custom applications can identify potential code base candidates for reuse. To produce a reusable software library, the development teams need to generalize software classes, refine the code so it stands alone or has minimum dependencies, and document the interfaces. Categorizing code by functionality and design patterns enables the development team to decompose new problems and match the functions and patterns.

An example of an application architectural design pattern is the Model View Controller (MVC), first described by Trygve Reenskaug in 1979. In the MVC pattern, the user interface is isolated from the business logic so that either can be worked on without affecting the other. Applications based on the MVC pattern can be adapted to new business logic or redesigning the user interface for different communities.[9]

Rapid Prototyping

Integrated Development Environments (IDE) and Rapid Application Development (RAD) frameworks enable prototyping and early demonstrations that engage customers. An IDE offers editors that color coding and completion, which reduces syntax errors. Other features include code libraries and integrated debuggers. A RAD offers graphical user interface design tools, code generators, and a framework of folders and libraries. Applying these concepts of reusable code, design patterns, IDEs and RADs to application development enables the development team to focus on the unique requirements that derive from an organizations process, data flows, and products.

FEDERAL ENTERPRISE ARCHITECTURE DATA REFERENCE MODEL

The Federal Enterprise Architecture (FEA) standard provides a Data Reference Model (DRM) that enables information sharing and reuse across the federal government; this standard facilitates description and discovery of common data and promotes uniform data management. The DRM abstract model is an architectural pattern to optimize agency data architectures. Standardization areas of the DRM focus on:

- ***Data Description*:** Provides a means to uniformly describe data, thereby supporting its discovery and sharing.
- ***Data Context:*** Facilitates discovery of data through an approach to the categorization of data according to taxonomies. Additionally, enables the definition of authoritative data assets within a Community of Interest (COI).
- ***Data Sharing:*** Supports the access and exchange of data where access consists of ad-hoc requests (such as a query of a data asset), and exchange consists of fixed, re-occurring transactions between parties. Enabled by capabilities provided by both the Data Context and Data Description standardization areas.[10]

COMMON INFORMATION MODEL STANDARDS

The Distributed Management Task Force (DMTF), an industry organization, established the Common Information Model (CIM) as an object oriented architecture for depicting and tracking complex interdependencies and associations among software objects. Interdependencies include logical network connections, physical devises, transactions, and database servers. A specification and schema constitute the CIM, which define details for integration with other management models, and actual model descriptions. CIM is a common data model of an implementation-neutral schema for describing overall management information in a network or enterprise environment.[11] Examples of CIM objects include Database, Device, Event, Security, Metrics, Network, Policy, System, Support, and User. Organizations can use the CIM as templates and guidelines for defining the data structures and integrating the schemas of the applications within their enterprise architectures. Vendors can extend CIM's common definitions to exchange semantically rich management information between systems throughout the network.[12]

CONCLUSIONS

Concepts discussed in this paper include:

- Common organization capability needs
- Case studies about data base development activities to define customers, suppliers, tasks, products, and data flows, and potential applications of a data exchange agreement
- Business process modeling methods, tools, technologies, and Kaizen events
- Model driven and service oriented architectures and a defined set of services
- Maturation of agile software development processes, code libraries, design patterns, and rapid application development
- Government and Industry data definition standards.

Weaving these concepts into an enterprise architecture integration plan and a documented repeatable process involves the following steps:

1. Establish procedures for collect, model, and graph processes and data.
2. Establish a development environment with IDE, RAD tools, and code libraries.
3. Identify interdependent communities.
4. Create process models that identify customers, suppliers, data flows, tools, storage, and security levels; apply BPMN, DFD, and other notations.
5. Formalize DEAs where necessary.
6. Identify common needs and define service requirements to meet the needs.
7. Develop web services, document interfaces, and deploy code libraries.
8. Identify data context, structure, types, flows, and frequencies; apply UML, entity relationship and other notations.
9. Map the data to a central WBS or PBS and map the data structures to the CIM
10. Determine the strategic processes and conduct Kaizen events to optimize them.
11. Develop BPEL code to orchestrate the web services to integrate applications.
12. Apply the CMMI to mature this process.

REFERENCES

1. An Assessment of the Degree of Implementation of the Lean Aerospace Initiative Principles and Practices within the US Aerospace and Defense Industry, Thomas E. Shaw, Alexander Lengyel, Greg Ferre, pg.44, 47, February 2004 http://dspace.mit.edu/handle/1721.1/7320
2. Object Management Group, Business Process Modeling Notation, last modified August 15, 2008 http://www.bpmn.org/
3. Web Service Orchestration with BPEL, Christoph Schittko, XML Conference and Exposition, December 7-12, 2003 http://www.idealliance.org/papers/dx_xml03/papers/04-06-01/04-06-01.html
4. GraphViz, released by AT&T under the Common Public License, December 11, 2004 http://www.graphviz.org
5. An introduction to Model Driven Architecture Part I: MDA and today's systems, Alan Brown, 17 Feb 2004 http://www.ibm.com/developerworks/rational/library/3100.html
6. What Is Service-Oriented Architecture, Hao He, September 30, 2003 http://www.xml.com/pub/a/ws/2003/09/30/soa.html
7. Sidebar: Watts Humphrey on Software Quality Software quality guru provides advice on implementing the Capability Maturity Model, Gary Anthes, March 8, 2004 http://www.computerworld.com/softwaretopics/software/story/0,10801,90799,00.html
8. CMM vs. CMMI: From Conventional to Modern Software Management, Walker Royce http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/feb02/ConventionalToModernFeb02.pdf
9. Model View Controller, Wikipedia, Last modified on August 15, 2008 http://en.wikipedia.org/wiki/Model-view-controller
10. Federal Enterprise Architecture Data Reference Model, Version 2.0, November 2005 http://www.whitehouse.gov/OMB/egov/documents/DRM_2_0_Final.pdf
11. Computer Information Model Tutorial, Distributed Management Task Force and WBEM Solutions, Inc., Copyright 2002-2003 http://www.wbemsolutions.com/tutorials/CIM/cimtutorial.pdf
12. Common Information Model (CIM) Standards, Distributed Management Task Force, Inc., June, 20 2008 http://www.dmtf.org/standards/cim/

IAC-08-D3.4-E5.4.5

MANAGING LARGE SCALE PROJECT ANALYSIS TEAMS THROUGH A WEB ACCESSIBLE DATABASE

Presenter: K. Bruce Morris
Manager, Exploration & Space Operations
Programs & Projects Office,
Marshall Space Flight Center,NASA
Bruce.Morris@nasa.gov

Author: Daniel A. O'Neil
Aerospace Engineer,
Marshall Space Flight Center, NASA
daniel.a.oneil@nasa.gov

Presentation Contents

- Common Capability Needs
- Composition of an Analysis Community
- Interdependent Use Cases
- Reports for Three Organizational Levels
- Integration Technology Concept Map
- Conclusions: An Enterprise Architecture Process

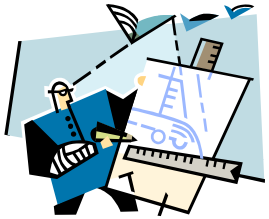
Common Capability Needs

- ***Project Management*** – Coordinating budgets, personnel, schedule, and products to meet requirements and mitigate risks
- ***Product Management*** – Transferring, storing, translating, and configuring within the context of a product breakdown structure that spans all levels of a program
- ***Process Management*** – Defining workflows, enabling approvals and concurrence, changing a products status, and notifying participants via e-mail
- ***Archiving*** – Aggregating designs, software, and data files into configurations and or compressing collections into files with indices and product descriptions
- ***Collaboration*** – Capturing comments, markups, revisions, discussions, and teleconference support
- ***Searching and Filtering*** – Finding information via key words, synonyms, and context within indices, product structures, and semantic information models
- ***Report Generation*** – Producing tables, spreadsheets, schedules, and diagrams that present data and relationships among people, products, and processes
- ***Information synthesis*** – Integrating data from multiple sources, plotting data, creating visualizations, and displaying results via portals and dashboards

Composition of the Analysis Community

Study Managers

- Establish study collections such as a Integrated Design and Analysis Cycle (IDAC).
- Select organizations, teams, Task Description Sheets (TDS), data
- Define ground rules and assumptions for the studies and identify review boards



Process Facilitators

- Identify milestones within the life-cycle of a TDS
- Identify roles involved in moving the TDS through its life-cycle
- Determine whether those roles concur or approve the TDS to promote it to the next milestone
- Create approval paths to be selected by TDS authors

Task Description Sheet Authors

- Describe the task objectives and assign the TDS to a study collection
- Link risks to be mitigated or requirements to be fleshed-out by the tasks
- Link initialization data and identify task products
- Specify dates for products and milestone dates

Discipline Specialists

- Conduct activities described in the TDS and produce data files, which they upload to Windchill
- Describe products in data records and record the Windchill hyperlink to the actual files
- Participate in the reviews of TDS and receive notification when a TDS of interest changes

Organizational & Project Managers

- Review, concur, or approve a TDS that affects personnel in their organization or work package
- Generate status reports of tasks performed by their organization
- Generate traceability and coverage reports that identify risks & requirements analyzed in tasks
- Generate traceability of models and simulations, used in tasks, to Constellation requirements

Interdependent Use Cases

Community

Goals

Study Managers

Assign Teams & Organizations

Establish Study Collections

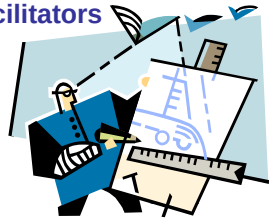
Define Ground rules & assumptions

Identify Requirements Documents

Generate Traceability & coverage reports

Status Reports

Facilitators



Identify Milestones & roles

Create Approval Path templates

Generate Model & Sim Traceability Reports

Generate Schedules

Schedules & Process Reports

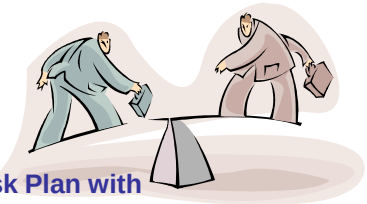
Task Leaders

Describe Tasks & Select path

Identify Needed Data

Link Models & Simulations

Link Risks & Requirements



Task Plan with Relationships to Risk & Req.

Specialists

Identify Models & Simulations

Produce data

Store data In Windchill

Describe data & link to Source files

Linkable published data

Org. & WBS Managers

Build Teams & Organizations

Approve or Concur on TDS

Assign Responsibilities

Generate Organizational Status reports

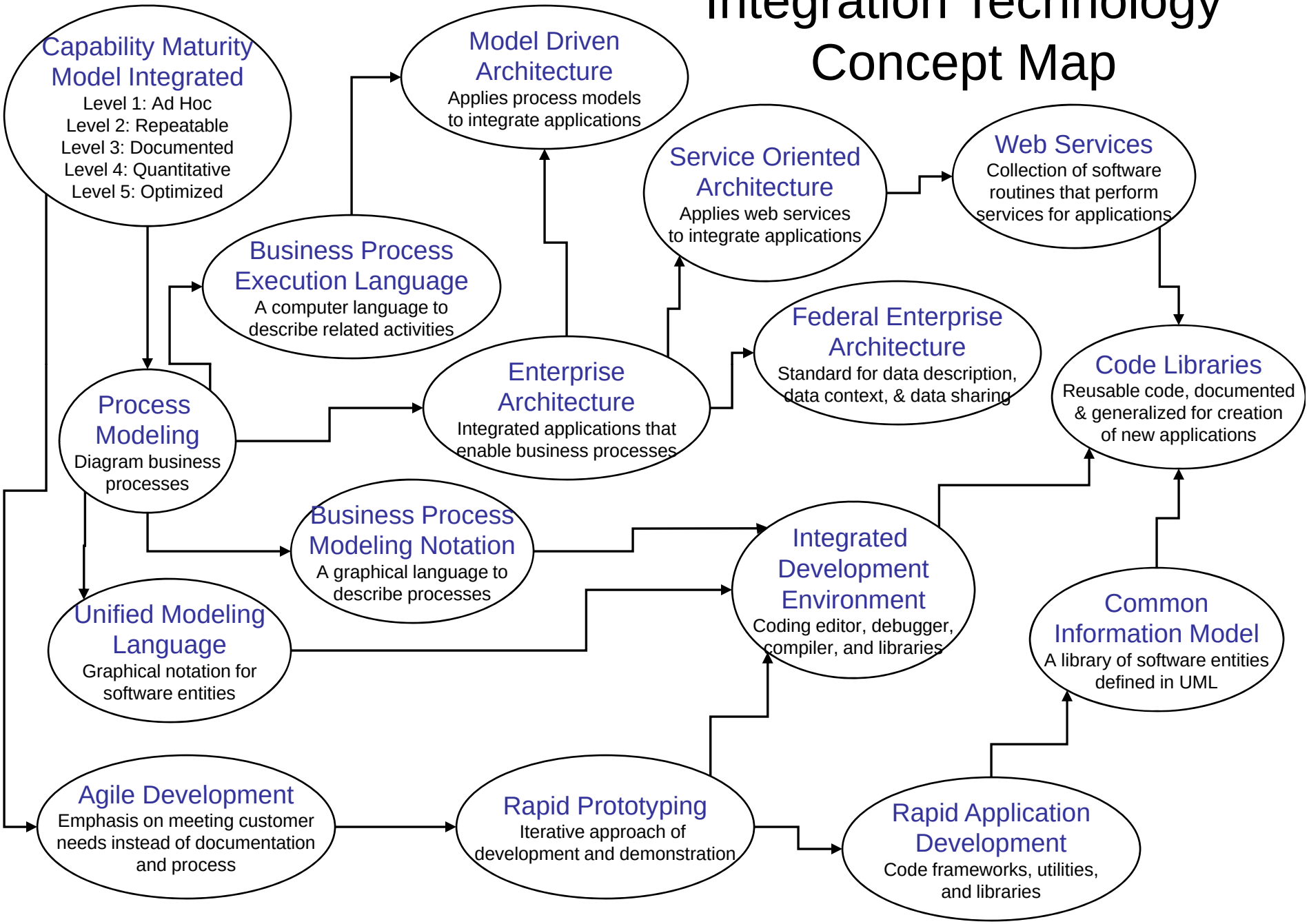
Relational reports about organizations and products

Generated Reports

Reports generated by the Constellation Analysis Integration Tool (CAIT) Database support three levels of NASA organizations.

- Level 1, Headquarters
  - Summary Task Description Sheet (TDS) status reports
- Level 2, Constellation Program
  - Detailed TDS status reports
  - Requirements coverage reports that map TDS to requirements
  - Risk coverage reports that map TDS to risks
  - Models and Simulations traceability reports
- Level 3, Project Offices
  - Task Description Sheet with signature blocks
  - Detailed product schedules compatible with MS Project
  - Excel spreadsheet export for detailed analysis

Integration Technology Concept Map



Conclusions

A Process for Enterprise Architecture Integration

1. Establish procedures for collect, model, and graph processes and data.
2. Establish a development environment with IDE, RAD tools, and code libraries.
3. Identify interdependent communities.
4. Create process models that identify customers, suppliers, data flows, tools, storage, and security levels; apply BPMN, Data Flow Diagrams, and other notations.
5. Formalize Data Exchange Agreements where necessary.
6. Identify common needs and define service requirements to meet the needs.
7. Develop web services, document interfaces, and deploy code libraries.
8. Identify data context, structure, types, flows, and frequencies; apply UML, entity relationship and other notations.
9. Map the data to a central WBS or PBS and map the data structures to the CIM
10. Determine the strategic processes and conduct Kaizen events to optimize them.
11. Develop BPEL code to orchestrate the web services to integrate applications.
12. Apply the CMMI to mature this process

Back-Up Charts

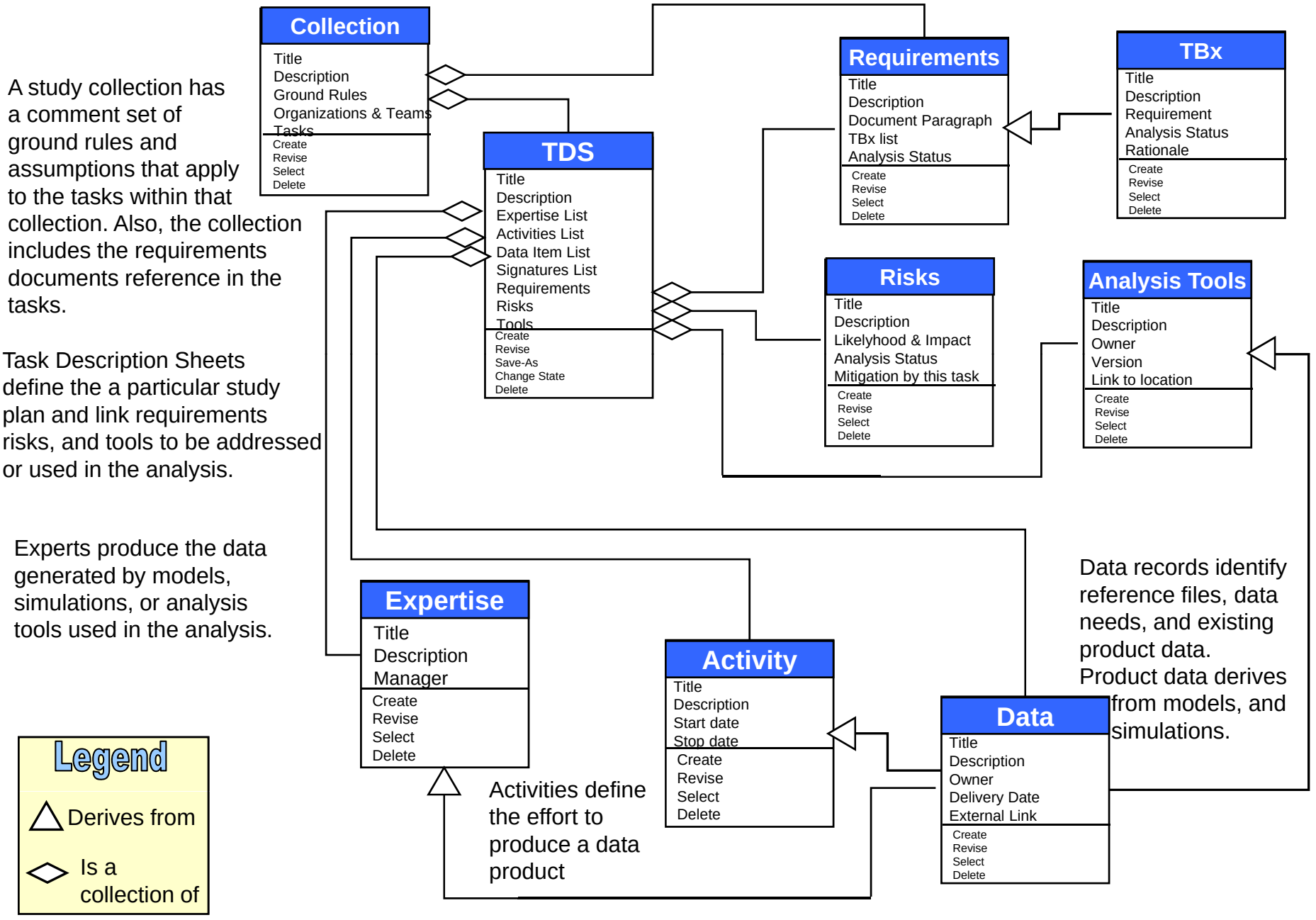
Data Classes in the CAIT Database

Task Description Sheet Related Classes

A study collection has a comment set of ground rules and assumptions that apply to the tasks within that collection. Also, the collection includes the requirements documents reference in the tasks.

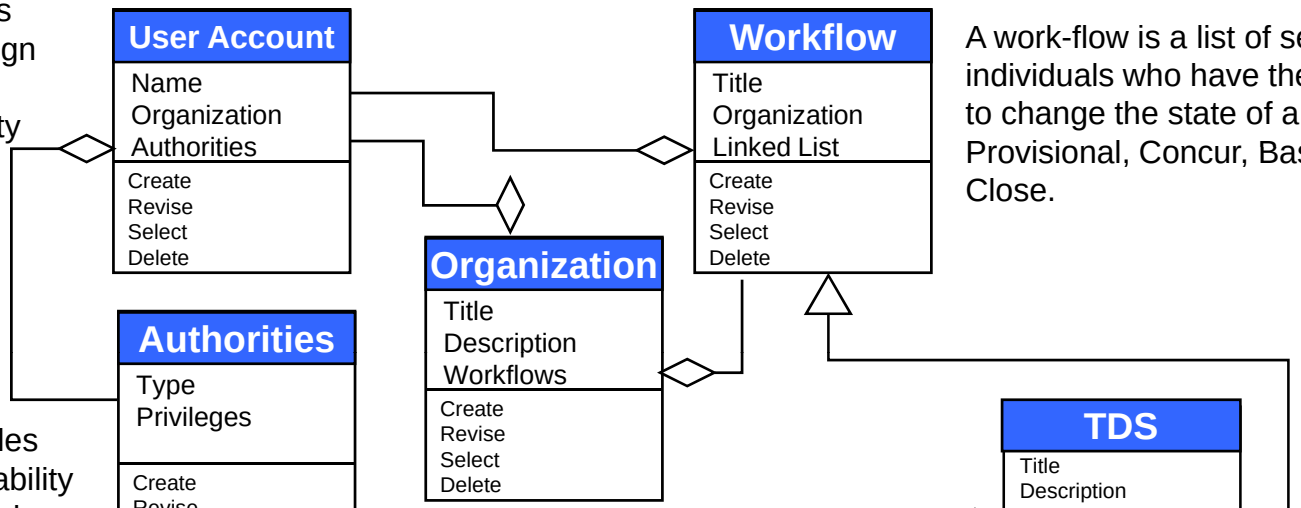
Task Description Sheets define the a particular study plan and link requirements risks, and tools to be addressed or used in the analysis.

Experts produce the data generated by models, simulations, or analysis tools used in the analysis.



Analysis Process Management Classes

Everyone starts with the Draft authority. Administrators and managers can assign additional authority. Certain roles or authority have the privilege to build a Workflow.



A work-flow is a list of selected individuals who have the authority to change the state of a TDS to Provisional, Concur, Baseline, or Close.

A new authorities or roles table provides the capability To define privileges to change the state of a TDS, create a work flow, or assign authority to an individual.

Each Expertise record has a manager with the authority to commit the time of the experts.

Schedulers and Project Managers create Activity records.

Legend

△ Derives from

◇ Is a collection of