Conceptual Data Model Evolution in Joint Strike Fighter Autonomic Logistics Information System of Systems Engineering

Tod Hagan
Sr. Systems Engineer
Modus Operandi, Inc.
thagan@modusoperandi.com

John Walker
Sr. Systems Engineer
NAVAIR, Patuxent River
John.Walker@navy.mil

Abstract

The complexity of modern systems engineering projects has dramatically increased over the past two decades. One reason for this complexity is the challenge of developing a complete and correct conceptual system data model. Systems being developed must use information from an increasing number of information sources including legacy systems, GOTS, COTS and custom databases. Conceptual system data models must constantly evolve and be verified to accommodate system enhancements or COTS package upgrades. The task of verifying conceptual model correctness is largely a manual task with limited tools. This paper describes current research to develop tools and techniques which allow systems engineers to perform comparative analysis between the conceptual data model for a new system and the physical data models represented in applications, databases and related systems. The Joint Strike Fighter (JSF) Operations and Support budget will be reduced from two thirds to one half the total life cycle cost [JSFSDD02]. An efficient and affordable JSF maintenance program is critical to the overall project success and ultimately the war fighter. This research is directed specifically to JSF but has broad applicability to any project with requirements to integrate existing information systems while maintaining a consistent enterprise model.

Key Words: Joint Strike Fighter Autonomic Logistics Information System, Enterprise Model Data Discovery, Conceptual Model Comparative Analysis, Conceptual Model Verification

1.0 Introduction

Currently, it is very difficult for systems engineers to perform a comparative analysis of data models used by the many different applications, databases and related systems that comprise their projects. This is a fundamental systems engineering task in the development of systems that rely on integrating information from legacy systems.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 JUN 2009	**N/A**	**-**

4. TITLE AND SUBTITLE
Conceptual Data Model Evolution in Joint Strike Fighter Autonomic Logistics Information System of Systems Engineering

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Modus Operandi, Inc.

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**13**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Specifically, the JSF must perform a comparative analysis of all the legacy logistics systems and the conceptual model in the Autonomic Logistics Information System (ALIS) JSF is developing. Automated tools are required to keep the ever-changing COTS and GOTS data sources aligned with the ALIS throughout the life cycle of the JSF. The difficult tasks in developing the automated support for analyzing, measuring or representing data model similarity include: What is the same? What is different? How can we display it meaningfully? For example, it may be that 90% of the fields for "part" tables in these databases are the same, but only 20% of the data types align properly. If two fields are strings, but one is variable length and one is fixed at 20 characters, how do we represent the comparison in a way that is meaningful to systems engineers? The problem increases non-linearly [LAM94] with the number of data sources (or applications) that must be compared and normalized on the JSF ALIS Project.

Current research is being done by NAVAIR and Modus Operandi, Inc. to develop new tools and techniques, which reduce the cost and risk of developing new conceptual system data models. Specifically, this research is focused on answering the following questions.

- How can relevant and meaningful data be discovered in existing information systems?
- How can the comparison of the target conceptual model to these information systems be automated and the comparison displayed in a meaningful and useful way?
- As the conceptual model and or the source data structures change, how do we revalidate the mappings between the information systems and the conceptual model?

Answering these questions is critical to maintaining a consistent, complete, and semantically meaningful data model for JSF ALIS.

The research questions have lead to the following research goals.

1. Develop an automated *physical data model discovery* capability;
2. Develop a *data model comparative analysis* capability;
3. Develop a *data model validation and verification* capability;
4. Integrate this *new capability with existing tools*

Goal 1, Automated physical data model discovery focuses on the ability to find data which may represent the conceptual data model.

Goal 2, Data Model Comparative Analysis focuses on providing engineers with the ability to compare two or more models in a meaningful way. This includes comparing two or more sources or comparing these models with a target reference model (enterprise model, ontology, unified warehouse model). Methods for measuring model similarity are significant aspects of this goal.

Goal 3, Data Model V&V, will focus on validating a source system data model against a conceptual model to ensure consistency and completeness, especially when changes are made to the canonical form or source data models. The objective here is to maintain consistency between source system data models and the canonical form as these evolve.

Goal 4, Integrate New Capability with Existing Tools, will focus on providing the new capability by extending leading data modeling tools, such as Borland's Together Control Center or IBM's Rational Rose. This will allow engineers to use "best of breed" tools for enterprise modeling and have new capabilities for validating those models against the conceptual model (enterprise model, ontology, etc.).

The research results to date for each of these goals are discussed in the following sections.

2.0 Automated Data Discovery

Since the beginning of the computer age, organizations have developed mission critical systems in a vertical fashion. Users use client systems and data sources developed together in a tight integration. The interface between the data and the client system, if documented, is mostly used to upgrade the client system. Today with the development of system of systems horizontal integration of those legacy systems are required. New systems need to push and pull data into those loosely documented data sources and interfaces. Consequently the ability to discover, access or browse data in a heterogeneous systems environment is currently a manual process. Automated data discovery is a challenging problem because: the tight integration between legacy client system and data sources, the inadequate or nonexistent interface definitions, the legacy developer knowledge loss and lastly, the security and information assurance issues of going around the client system directly to the data sources.

The ability to discover data in legacy systems is vital to JSF. The eight partner nations, Australia, Britain, Canada, Denmark, Italy, the Netherlands, Norway, and Turkey all have different logistic support systems. To support prognostics for the DOD, JSF ALIS will require historical data from legacy systems such NALCOMIS, DeckPlate and CAMS [JSFALISPDR04]. Britain has its own maintenance support system; the Logistical Information Technology System (LITS). Currently, there is no tool to discover and compare data in these and other legacy systems to the ALIS conceptual model.

Our research has shown two scenarios for constructing information models and performing comparative analysis. The first scenario, a top-down approach is being used by JSF ALIS. The second scenario, a bottom-up approach is being used by the USAF 45th Space Wing on their Knowledge Management Initiative.

Top-Down Scenario – JSF ALIS

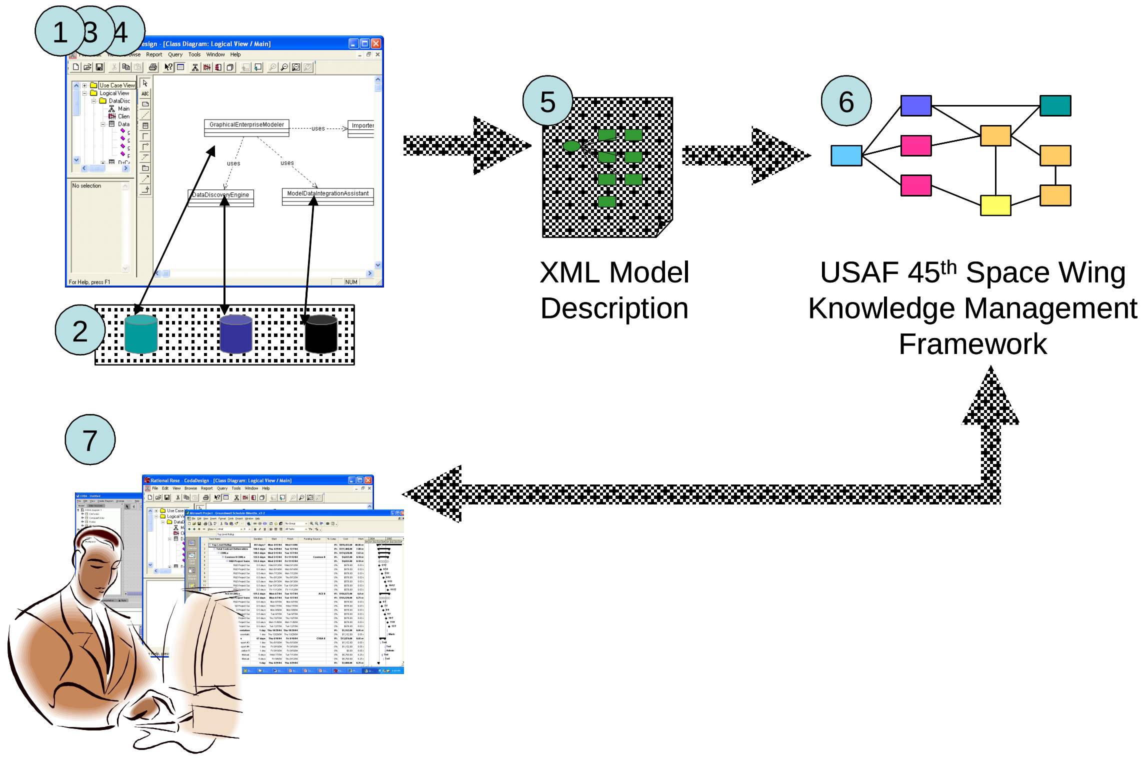
In the top-down scenario, the conceptual model is created first using modeling tools such as IBM's Rational Rose, Argo UML or Borland's Together Control Center. The conceptual model is then reviewed and approved by cognizant project stakeholders. The systems engineer then discovers physical data sources and maps them to the approved conceptual model. Once the mappings have been identified, the systems engineer can perform a comparative analysis to identify where the conceptual and physical models are out of sync. The conceptual-to-physical model mapping is persisted so that at any time in the future, the comparative analysis can be re-run to verify the conceptual model.

Since the emphasis of this paper is on JSF ALIS, additional details about the top-down scenario and comparative analyses are included in later sections of this white paper. Let's briefly look at data discovery in the bottom-up scenario being used by another DOD organization.

Bottom-Up Scenario – USAF 45th Space Wing

In the bottom-up scenario, physical data models are discovered and reverse engineered to create the conceptual model. The 45th Space Wing also requires the capture of meta data in the conceptual model about the information system being discovered. Examples of their meta data include the type of units for a given attribute, or data source owners and contact information. The bottom-up scenario is illustrated in the figure below.

This data discovery capability by itself has many potential applications to reduce the cost and risk of developing new systems. One advantage is simply the ability to create a complete model of existing enterprise information systems. Once the logical model has been created and the mapping to physical data sources persisted, the logical model can be exported to standard formats. For example, the USAF 45th Space Wing is using this technology to build and enterprise model description and then export it to their Knowledge Management Initiative.



In step 1 of the scenario, the modeling tool is started. Section 5 discusses the integration of this capability with modeling tools. In step 2, existing information sources are discovered and previewed. The discovered information sources are reverse engineered (step 3) to create the conceptual model. Next, relations between reverse engineered models are created (step 4). In step 5, a conceptual model description (e.g. XML) is exported to a form usable by the 45^{th} SW KMI. The KMI imports the model description in step 6 to configure the data integration layer. The KMI data integration layer contains servers which access information from physical data sources and provide a standard interface (e.g. web services) to new client applications being developed (step 7).

Data Source Filtering

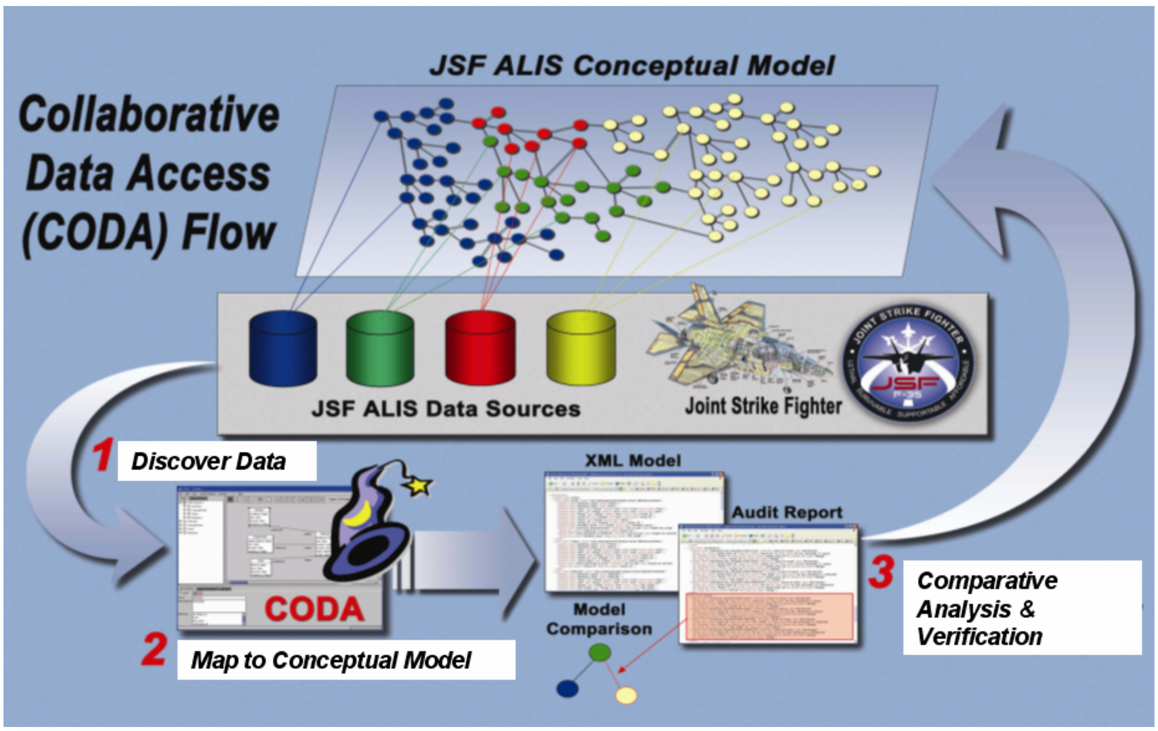
Our research indicates that most information systems contain information that will not be relevant to the conceptual model being developed. For example, a typical Oracle database has over 1000 system tables used to mange its configuration. These tables are typically not important to the systems engineering that is creating a conceptual model. Domain specific or target project dictionaries can be used to efficiently discover relevant data in information systems. This reduces the effort by systems engineers to find the correct conceptual model mapping. This domain intelligence can be re-used on similar projects.

Follow-on Research

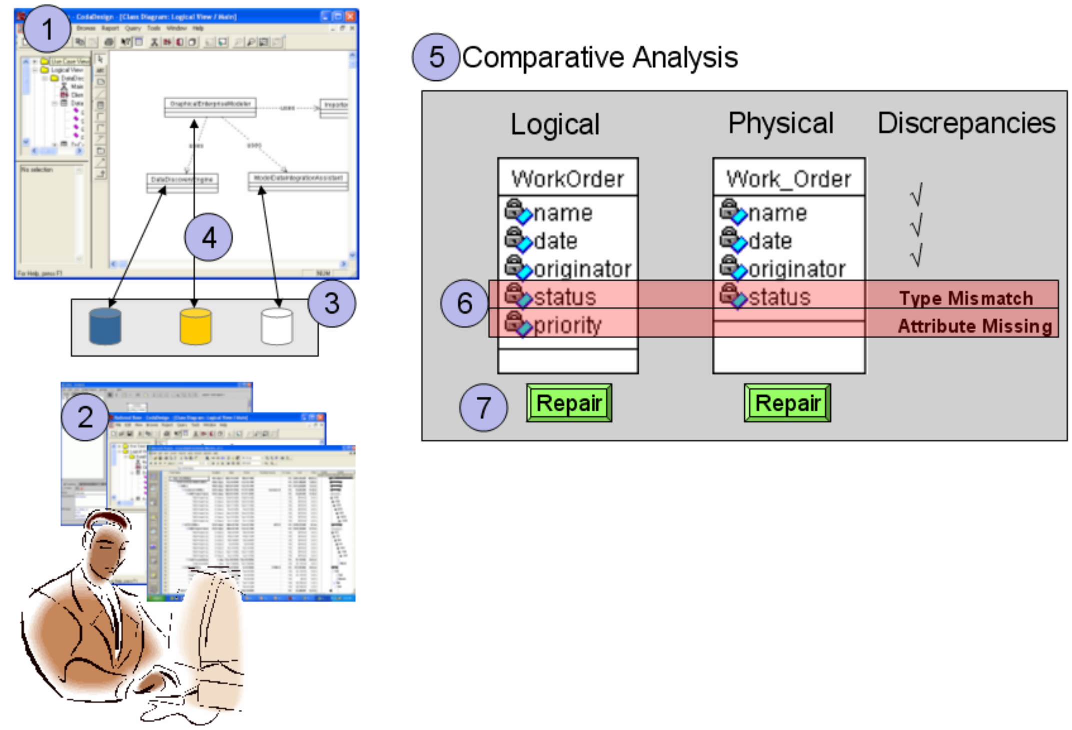
An interesting topic for follow-on research would be to investigate the feasibility of automated logical model creation starting with system requirements or use cases. As a result of processing the system requirements or use cases, the tool could generate a 0.1 version of the logical model.

3.0 Data Model Comparative Analysis

Three primary tasks are performed in the comparative analysis process. Before these tasks begin, the conceptual model is created using common modeling tools such as IBM's Rational Rose or Borland's Together Control Center. The first step is to discover the physical data sources that are represented in the conceptual model. The second step is to designate or map the conceptual model to a physical model or existing information systems. Comparative analysis of data models is complicated by the requirement to "map" each application's data source into a JSF conceptual data model (JSF ontology, enterprise model or canonical form). The third task is to perform the actual comparative analysis to verify the conceptual model. The following figure depicts sequence of these tasks in the context of JSF ALIS.



Now that we've seen the flow at a high level, let's look at aspects of comparative analysis in more detail. In the JSF ALIS top-down approach, the systems engineer will need to compare the conceptual model to one or more candidate physical models. The systems engineer needs to perform "what-if" scenarios by comparing several physical models for "best fit" or "most similar". The JSF ALIS scenario is illustrated in the following diagram.



The steps for this top-down comparative analysis are as follows:

1. Conceptual model created in IBM's Rational Rose, Borland's Together Control Center, etc.
2. JSF ALIS applications, CSCIs and the physical data models are developed based on approved conceptual model.
3. While viewing the conceptual model, data sources are discovered and previewed
4. The conceptual model is mapped to physical data sources.
5. The logical and physical models are compared. Does the "as built" model match the "as designed" model?
6. Model discrepancies reported to the user. A "Type Mismatch" as indicated in the figure could be that one model attribute is a string while the other is an integer. Another example shown is the case where there is no matching attribute in the physical data source.
7. This step gives the user the ability to repair discrepancies, either the conceptual or physical data models. This assumes the user has the proper authorization by following the projects change process, or control over the physical data model (which is unlikely in the case of COTS products).

A method for similarity measurement evaluates and quantifies the similarity between two data models. This will be a "closest fit" model that is intended to provide an overall assessment of the "likeness" between two data models on different levels (schemas, fields, data types, etc.). For example, one candidate physical mapping may be 80% match while a second candidate may be only 20% match in terms of its attributes.

4.0 Data Model Verification and Validation

Conceptual data model **verification** seeks to answer the following question; as currently mapped, does the conceptual data model still match the physical data model? The JSF ALIS is a hybrid solution composed of COTS and custom ALIS (e.g. JSF Prognostic Health Monitoring) schemas. The COTS products will likely include Seibel, MXI Technology, and a supply chain management product (currently I2, but this may change to Honeywell, IBM, or other). An automated capability is needed to compare and verify the ALIS conceptual data model to both COTS and ALIS schemas. This is currently a manual process and there are literally thousands of tables. To make matters even more challenging, the conceptual and physical data models will evolve over time and thus increasing potential to be out of sync. There are many reasons for this including:

- COTS upgrades may change how their data is physically represented
- Upgrades or enhancements to the system resulting in schema changes
- Conceptual data model evolution as a result of system upgrades and enhancements
- New information integration activities

Once an out-of-sync condition is detected, the ability to repair the model could prove useful. Repair of the conceptual data model and custom schemas are possible. Obviously, repair of the COTS schemas would not be possible. When repairs are made, it is assumed the user has the authority and has checked out or locked the model in the designated configuration management tool.

The JSF ALIS engineering team also needs the ability to estimate the impact of a change (impact analysis) to physical or conceptual data models. For example, if a change is made to a specific class or a COTS vendor changes their model, what are the impacts? The issue of impacts caused by changes to COTS is a very interesting one beyond the scope of JSF ALIS. As more and more projects are mandated to use (and therefore become heavily dependent upon) COTS, the issue of analyzing and quantifying the impact of COTS changes to the project/application is essential. Coupled with the average lifespan of a given COTS package version, which is approximately 4 years according to JSF ALIS engineers, any product/application with a longer lifespan will face upgrade cycles of varying effort and risk. Helping to assess that effort and risk is an essential need for any lengthy product lifespan.

Conceptual data model **validation** seeks to answer the following question; how does the current model compare to appropriate industry standards. In the case of JSF ALIS, the domain is DOD Aeronautical Logistics. There may be requirements for the conceptual model to be compliant with industry reference models. Reference models such as SCOR and ISO 11179 will certainly be of interest to JSF ALIS.

5.0 Integrate Capability with an Existing Modeling Tool

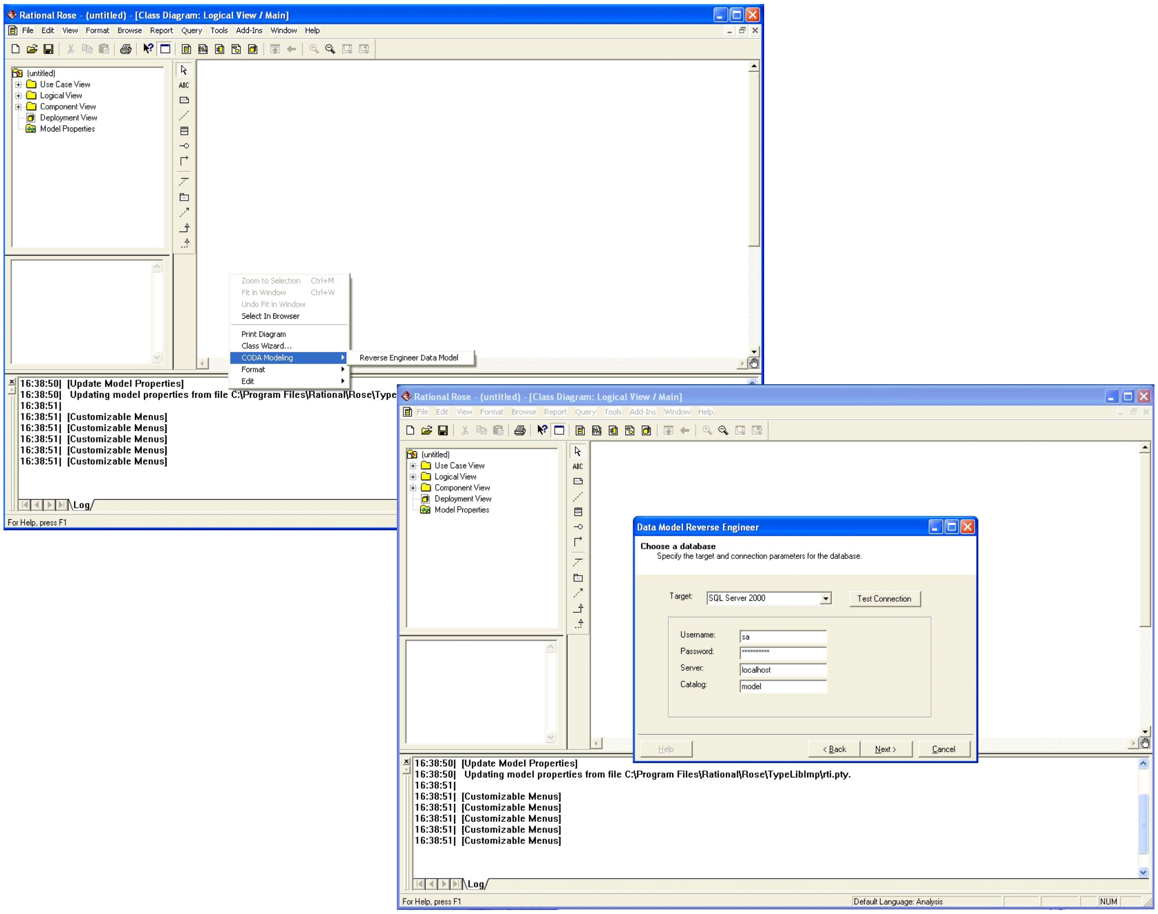
A key success criterion of our research is that the resulting tool or capability is easily adopted by end users. The optimal approach is integration into the existing system development processes, avoiding the need for systems engineers to learn a new tool. The initial version of the tool was built as a standalone application. While this worked well and served as good proof-of-concept, it was a different tool as compared to those already being used by JSF ALIS systems engineers. JSF ALIS systems engineers were using IBM's Rational Rose to create the conceptual model. We explored the concept of integrating the standalone tool's capability into Rational Rose. This idea was presented to and embraced by JSF ALIS systems engineers. This was a significant change to our implementation approach, but had many benefits.

By integrating new capabilities into an exiting COTS modeling tool, we are able to leverage many of the COTS features. Developing a graphical rendering engine which is capable of efficient model layout is very time consuming. Computer languages such as Java and C# sharp provide many components but the program developer still has to solve layout or positioning tasks. Another COTS tool feature which can be leveraged is the ability to reverse engineer data sources. Both Rational Rose and Borland Together have the capability to reverse engineer common data sources such as ODBC databases. For our purposes, the applicable features can be summarized as follows:

- General enterprise data model creation
- Limited data source reverse engineering
- Limited model export capability
- Limited model comparison

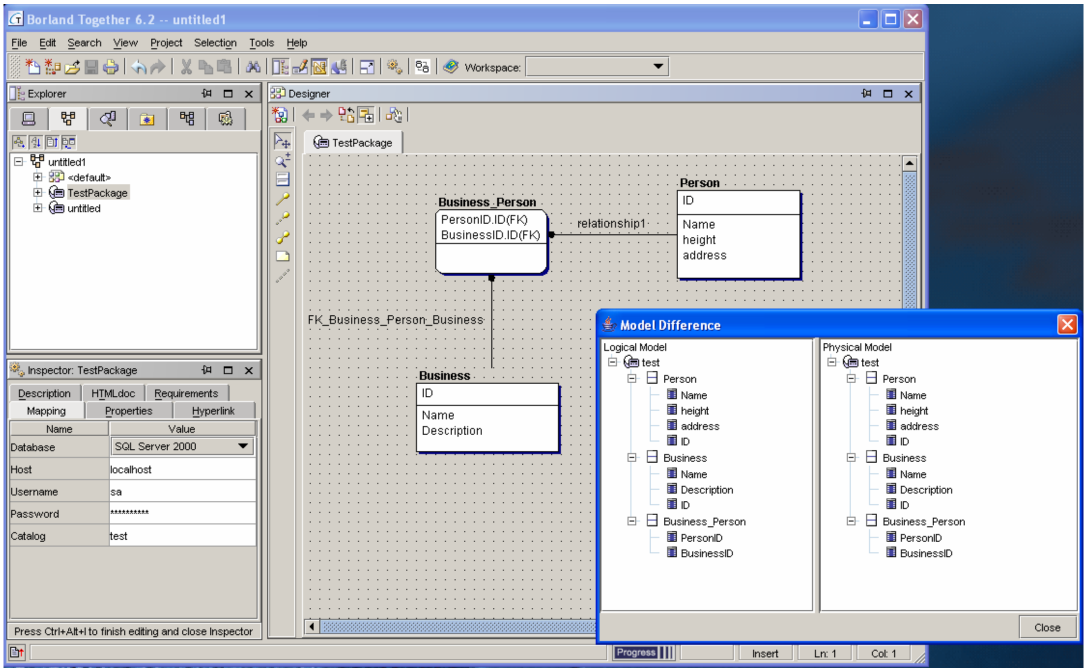
COTS Modeling Tool Extensibility

Both Rational Rose and Borland Together provide an API to extend their functionality. Initially the JSF ALIS team used Rational Rose to develop their conceptual data models. We determined that Rational Rose could easily be extended by creating a Component Object Model (COM) add-in or object. The COM add-in is typically created in C# and automatically loaded at run time. The following screen shots show Rational Rose extended to connect to reverse engineer a SQL Server Database. In the first screen shot, a new menu item was added allowing the user to reverse engineer a data source. In the second screen shot, a database selection dialog has been created. The user has selected a SQL Server 2000 Database and entered username and password information. Once connected to the database, the user can browse information and create conceptual data models (bottom-up scenario) or map to existing conceptual data models (top-down scenario).

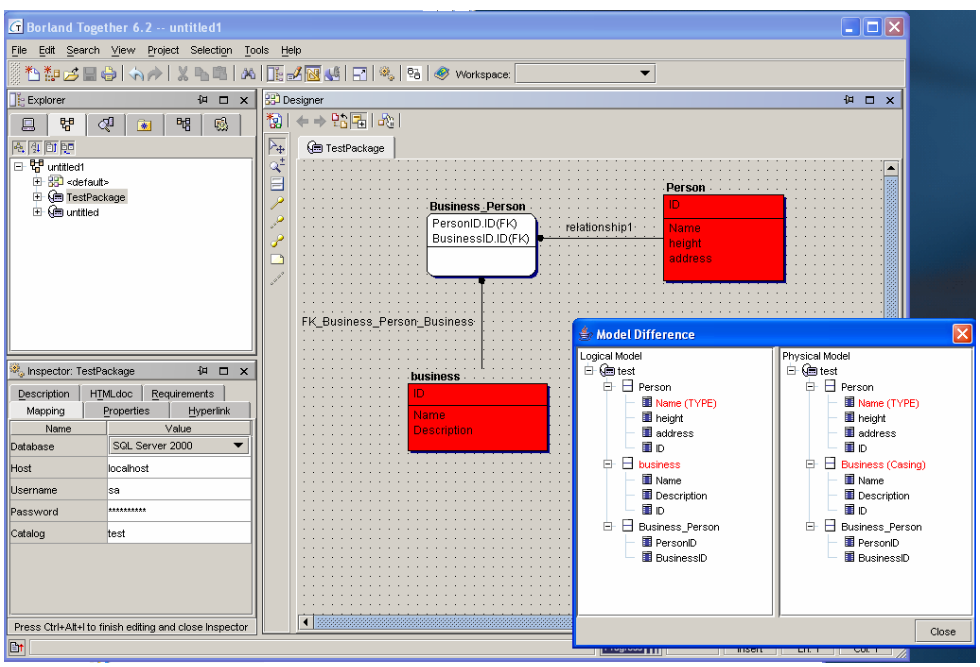


Recently the JSF ALIS team changed conceptual modeling tools from IBM’s Rational Rose to Borland’s Together Control Center. Borland Together has a Java API which provides access to almost all of its features making a very tight integration possible. While both Rational Rose and Borland Together provide an extensive API to extend their capabilities, Borland Together provides access to a much deeper level (any non–private attribute is accessible). The implications of this are that more of the tool’s features can be leveraged and more seamless capability integration is possible.

The following screen shots show a prototype integration with Borland Together. In the first screen shot, a logical model has been created and mapped to a physical data source. Information associated with the mapping of a specific class is shown in the lower left frame. The physical data source and connection information can be edited in this frame. To invoke the comparative analysis capability, the user simply right clicks on the diagram and selects “Compare”. When a discrepancy is found, the class will be highlighted in red on the class diagram. To see comparison details, the “Model Difference” dialog is invoked by right clicking on a class and selecting the “Compare/Repair” difference menu option. The resulting dialog shows a side by side comparison of the two models in detail. The dialog indicates there are no differences between the logical or physical data models.



The second screen shot shows that discrepancies have been found in the data model. The "Person" and "Business" classes are highlighted on the class diagram. This quickly and intuitively shows the user that a problem in the data model has been found. By highlighting a class with discrepancies, the user will be able to zoom out on a large model and easily find were there are problems. To see specifically see where the problem is, the user right clicks on the class and selects "Compare". Model discrepancies in the "Model Difference" dialog are also identified in red. The comparative analysis capability has detected differences in the "Person" and "Business" model mappings. The "Person" class has a type mismatch and the "Business" class has a case mismatch.



The capabilities we are continuing to investigate are listed below.

- *Mismatches of Interest:* The capability to specify the types of conceptual vs. physical data model mismatches which are of interest.
- *Model Repair:* The capability to repair either the physical or conceptual data model when discrepancies are found.
- *Candidate Best Fit Model:* A best fit capability which automatically searches data sources for a physical data model that most closely matches the logical data model.
- *Domain Dictionary:* A domain dictionary to support an intelligent physical data searches.
- *Extended Reverse Engineering:* The capability to reverse engineer other, potentially non-standard data sources. As JSF ALIS evolves and prognostic maintenance capabilities added, information from data sources such as NALCOMIS, DeckPlate and CAMS will need to be analyzed. LITS from the United Kingdom will also need to be analyzed.
- *Impact Analysis Capture:* The ability to capture the amount of impact to access the cost of a change to the conceptual or physical models.

6.0 Conclusion

The maintenance challenges and associated costs of the Joint Strike Fighter (JSF) Program, DOD's largest acquisition program to date, will be unprecedented in aeronautical history. At the heart of these challenges is designing, developing, validating and maintaining JSF's unified enterprise model and keeping it consistent with the data sources it accesses. In the JSF environment, information failure can lead to mission failure.

Our research to date has demonstrated the feasibility of developing tools and techniques for developing JSF's conceptual model, integrating it with existing data sources, and maintaining consistency among these models in the face of these dynamically changing structures. Our initial data discovery capability has been demonstrated using several operational military data sources, without modification or simplification, and has greatly improved our engineers' ability to map those sources to an evolving conceptual model. This has been demonstrated for both top-down data engineering, starting with the conceptual model and mapping to data sources, and for bottom-up data engineering, beginning with the data sources and synthesizing a common conceptual model from them. We have successfully integrated these techniques into two commercial design products mitigate the problem of stovepiped applications and the data they produce. We have extended these products to support side-by-side comparison of data sources and conceptual data models, facilitating the identification of discrepancies to support the evolutionary engineering required by JSF. Finally, we have demonstrated the capability to continuously validate consistency among the data models and verify whether changes requiring modification have occurred. As this capability is fully implemented, all indications are that it will greatly simplify the systems engineering tasks for JSF's enterprise modelers.

7.0 References

[JSFSDD02] Joint Strike Fighter SDD, Joint & International Interoperability Challenge Briefing, Lockheed Martin, September 2002

[JSFALISPDR04] Joint Strike Fighter Autonomic Logistic Information System Preliminary Design Review-1, Lockheed Martin, March 1, 2004 – March 4, 2004

[LAM94] LaMonica, Frank S. et al., "Automated System Engineering Automation (ASEA) for the 21st Century." National Council on Systems Engineering (NCOSE) Conference, July 1994.