ENSE 622: Systems Engineering Requirements, Design and TradeOff

Systems Engineering Tools

Mark Austin
E-mail: austin@isr.umd.edu

Institute for Systems Research, University of Maryland, College Park

- State-of-the-Art Capability
- IBM Telelogic Doors, IBM Telelogic Tau, IBM Telelogic Rhapsody.
- SLATE (System Level Tool for Enterprises)

Part 2. Experiments at UMD

- Lessons Learned/Present-Day Limitations
- Requirements Engineering and the Semantic Web
- PaladinRM. Visualization of Requirements Hierarchies

Acknowledgements

- Scott Selberg, Vimal Mayank, Natasha Shmunis, Parastoo Delgoshaei.
State-of-the-Art Capability

Requirements management systems such as SLATE and DOORS are ...

... implemented as monolithic systems, having centralized databases.

Emphasis is placed on:

1. Representation and traceability of requirements on the level of natural language sentences,
2. Database storage,
3. Support for multiple viewpoints,
4. Traceability between objects, and
5. Evaluation of requirements.
Requirements Organization

Organizing requirements into the right structure can help:

1. Minimize the number of requirements (i.e., eliminate redundant requirements).
2. Find sets of requirements relating to a particular topic.
3. Detect omissions, duplications and inconsistencies.
4. Eliminate conflicts among requirements.
Representation of requirements in DOORS
Schematic of a Traceability Report

- User requirements
- Technical requirements
- Design
- Test cases

Example:

3.1.2.3 Stopping

Users shall be able to stop safely.

The car shall be able to stop from 10 kilometers per hour to 0 kph in 2 seconds.

TRN-CS-35

FR-24

Disc brakes

TRN-TP-34

High Speed Braking Test

TRN-TP-35

Low Speed Braking Test

TRN-TP-34

High Speed Braking Test

TRN-TP-35

Low Speed Braking Test

TRN-TP-34

High Speed Braking Test
Latest Capabilities (as of March, 2011)

- Allows all stakeholders to actively participate in, and manage, the requirements process
- Powerful life cycle traceability to help teams align their efforts with the business needs and measure the impact that changes will have on everything from business goals to development
- Ability to exchange requirements data with other requirements management tools
- Test Tracking Toolkit for small-scale test environments
- Integrates with the enterprise architecture, product portfolio management, model-driven development, quality management, and change and release management solutions from IBM and third party companies.

Web site. Google: “IBM DOORS”
IBM Telelogic Tau

TAU is a UML/SysML modeling tool supporting automated code generation and model verification.

TAU is today made by IBM. It was part of the IBM acquisition of Telelogic 2008.
DOORS and IBM Rational Tau

- Role-based integration of Tau and DOORS provides systems engineers and software developers with...
  
  ... fine-grained traceability of their system and software models, easy link creation, and direct navigation between DOORS and Tau.

- From Tau, users have...
  
  ... full visibility of requirements in DOORS and can link model elements in Tau with DOORS requirements without opening DOORS.

- DOORS users can...
  
  ... readily access both the model and the links whenever needed, such as for traceability report generation and impact analysis.

- The components necessary for the integration are present within both products, allowing for fast, easy set-up at no additional cost.
IBM Rational Rhapsody

Rhapsody Visual Design Environment

Rhapsody is a visual design environment for ...

... requirements creation and modeling of embedded software.

Rhapsody enables user to accomplish the following tasks:

Analysis and Design

- Define system requirements, identify necessary objects, and define their structure/behavior through UML/SysML diagrams.
- Trace requirements to the design.

Implementation and Testing

- Automatically generate code from the analysis model, then build and run it.
- Simulate the application to perform design-level debugging.
Rhapsody Development Process

Model entry → Source code compiler and linker →
Model entry → Model compiler → Test conductor → Model-level execution tools → Model monitor → Rhapsody Desktop

Model animation

Platform-independent applications.
Platform-independent framework.
RTOS
Hardware
Application on Target

Application execution behavior
Rhapsody Development / Use Cases / Scenarios, .. etc

- Use Case Diagram
- Use Case 1
  -- scenario 1
  -- scenario 2
- Use Case 2
  -- scenario 3
  -- scenario 4
- Fix errors
- Sequence of tasks
- Activity Diagrams
- Individual Use Cases and Scenarios
- High-Level Requirements:
  - Req 1.
  - Req 2.
- Sequence Diagrams
- Models of System Behavior and System Structure.
- Assembly of Active Objects
- test conductor
- executable program
- source code
- code generation
- round tripping
Framework for Active Objects

Generic framework for development of active objects.
Active Object Behavior

Schematic for modeling of active object behavior.
Library for Source Code Generation

Real-time applications are created extensions to the predefined core Object eXtended Framework (OXF).
The OXF execution and event-handling framework corresponds to an assembly of objects instantiated from four classes: Active, Reactive, Event and EventQueue.

**Active Classes**

- An active object (UML definition) owns a thread and can initiate control activity.
- An active class is a class that owns a thread of execution and has event dispatching and timeout scheduling functionality.

**Reactive Classes**

- A reactive class is one that can react to events, i.e., it is an event consumer. It is represented in the framework by OXFReactive. Every reactive class has an associated event manager, which is is an active class.
Events and Operations

Events correspond to ...

... well-defined instantaneous occurrences that affect behavior of the class.

Operations correspond to ...

... services or functionalities provided by the class.

Rhapsody handles three kinds of events:

- **Signal Events**: Correspond to asynchronous stimuli communicating between instances.

- **Time Events**: Signal that a specified amount of time has elapsed since a state was entered. Time events are called timeouts.

- **Triggered Operations**: Are (synchronous) operations implemented by the immediate injection of an event. A triggered operation does not have an implementation method – instead it may be the trigger of transitions in the class statechart.
State Machines

- The run-time behavior of objects is defined by statecharts, and how they react to events and operations.
- Reactions can include transitions between states and possibly execution of some actions (embedded within a procedure?).

Key points:

- A state in the statechart is an abstraction of the mode in which the object finds itself.
- A message triggers a transition from one state to another.
- A message can be either an event or a triggered operation.
- An object can receive events it sends to itself (a self-message).

Statechart guards and actions can be defined in terms of operations and attributes of classes.
Benefits of Code Generation

Code generation extends visual modeling with UML/SysML by ...

... giving a developer visual building blocks but a way to generate small and fast code from these building blocks.

This ...

... eliminates repetitive and error-prone manual coding. Capsules, ports and protocols provide a convenient communication mechanism that would normally be hand-coded.

Round Tripping

Round tripping allows ...

... elements of source code (e.g., attributes and methods) to be automatically synched with the model.
Step-by-Step Procedure for Executable Model Assembly

- The global system structure is established as a set of components.
- The structure of each component is detailed by means of class diagrams.
- The behaviour of each class is specified using a state machine, where each state represents a stage in the lifecycle of a typical instance of the class.
- A transition rule specifies the new state achieved when an object in a given state receives a particular event.
- Each event represents an incident during the object lifecycle, as the reception of a method call, a signal, or the expiration of a timer.
- Transitions and states may have associated procedures (sets of actions) that model the behaviour executed when a class instance enters, stays in or exits a state.
- Procedures are specified using an action language.
Example 1. System Structure Modeled as a Class Diagram
Example 2. Use Case Diagram for a Simple Lamp
Example 2. Statechart Behavior for a Simple Lamp
Example 3. Broadcast Behavior to Multiple Destinations

New capability in Rhapsody 7.5.
DOORS and IBM Rational Rhapsody

This integration enables ...

... requirements to be traced through to systems and embedded software design expressed in UML or SysML models.

Requirements scenarios modeled in Rhapsody can be ...

... simulated to verify requirement accuracy.
High-Level Goals and Objectives

The goals of SLATE (...used by NASA Goddard; Northrop Grumman) are to

... help systems engineers manage the complexity of large multi-phased programs.

System descriptions are built from ...

... objects, arranged into hierarchies.

System behavior is ...

... described in terms of functional flow-block diagrams.

Applications are modeled as ...

... a network of connected requirements and design abstraction hierarchies.

Note. SLATE was initially developed in the early 1990s.
Abstraction Blocks

- Abstraction blocks are used to build things such as descriptions of systems, organizations, test systems, ... etc.
- The attributes of abstraction blocks are predefined text or numeric values that describe the object to which they are attached.

Requirements

- Requirements capture the intent of the product design.

Budgets

- Budgets relate measurable objectives (e.g., cost, weight, schedule, reliability, throughput) to a project design.

Notes

- Notes provide a project notebook capability for documenting decisions, rationale, ... other design issues.
Anatomy of a Generic Object
Hierarchical Organization

- SLATE (System Level Tool for Enterprises)

- Step 1: Define Requirements Hierarchies

- Requirements
  - Performance
  - Interface
  - Functional
  - Design
  - Constrained
  - Organizational
  - Acceptance

- Hierarchy 1
  - System
    - Subsystem_1
    - Subsystem_2

- Hierarchy 2
  - System
    - Subsystem_1
    - Subsystem_2

- Requirements Folder

- Hierarchy Folder
Traceability Links and Translational Mappings (TRAMs)

- Relationships define connectivity between requirements and abstraction blocks in the design hierarchy.
- Translational mappings (TRAMs) establish many-to-many relationships among abstraction blocks in different abstraction block hierarchies.

Elements of a translational mapping (TRAM)
Modeling of Translational Mappings (TRAMs) across Hierarchies

Mapping Across Hierarchies

- Requirements
- Functional Decomposition
- Functional Implementation Map (FIM)

- Electronics
- Mechanical
- Software
Modeling and Graphical Representation of TRAMs

- Part a shows a scenario with two source abstraction blocks and two destination abstraction blocks.
- Part b shows an outline view of the icons that will be displayed when the TRAM is expanded from one of the source abstraction blocks.
- Part c shows the icons that will be viewed when the TRAM is expanded about one of the destination abstraction blocks.
Abstraction block 1 complies with requirements R1 and R2,
Abstraction block 2 complies with requirement R1, but not R2,
Abstraction block 3 is defined by R3 – It also complies with R1 through the TRAM mechanism.
Example 1. Radar System (from the SLATE training materials ... )
Details of implementation in SLATE

(a) Assignment of requirements to abstraction blocks.

(b) Use a transitional mapping to link abstraction blocks in two viewpoints.
SLATE: Implementation

(c) Folder visualization of requirements in SLATE.

(d) Representation of abstraction block hierarchy and translational mappings (TRAMs) in SLATE.
Part 2. Experiments at UMD
Lessons Learned from Industry

Support for Team Development

• DOORS and SLATE both support a top-down approach to systems development. Tool support for the bottom-up integration of subsystems is missing!

Visualization

• DOORS and SLATE do not support domain-specific visualization of engineering systems.

• Now that Javascript and XSLT are here, this limitation can be mitigated.

Tools are Designed to be Process Neutral

• DOORS and SLATE are designed to be process-independent tools – that is, a company supposedly use the tool to enhance any high-level systems engineering process.

• Company training procedures focus on step-by-step procedures for low-level tasks.

• In practice, many employees have great difficulty in scheduling these low-level tasks to match organizational processes.
Lessons Learned from Industry

Requirements Engineering Work Breakdown Structure and Industry Toolset Weaknesses

How do we capture, represent and use knowledge through requirements engineering activities?
Lessons Learned from Industry

Thick Descriptions

- Current tools lack the ability to store informal representations (i.e., so-called thick descriptions) of systems conveying information along subtle or implied lines.

Model Driven Trace Capture and Usage

- Current tools lack mechanisms for "easy linking" of models into the design environment (e.g., SLATE).

Abstraction Mechanisms

- Current tools lack the ability to search and explore requirements at various levels of abstraction.

Inference Services

- Current tools are incapable of analyzing requirements for completeness or consistency. Search mechanisms are limited to keywords, which can be limiting for custom jargon in multidisciplinary and multilingual projects.
Layers of Abstraction and Technology in the Semantic Web

Layers of Abstraction

- Applications
- Ontology and reasoning layers
- Data layers
- Representation / syntax layers

Semantic Web Technology Stack

- Applications and Interfaces
- Trust
- Proof
- Unifying Logic
- Ontology: OWL
- Rules: RIF
- RDFS
- Data Interchange: RDF
- XML
- Unicode
- URI

Encryption
Mapping of Requirements (Creation Branch) to the Semantic Web Layer Cake

Semantic Web Layers

- Trust
  - Digital Signature
  - Logic
  - Proof
- Ontology Support
- Resource Description Framework + Schema
- XML - Structured documents
- Namespaces + XML-Schema
- Universal Resource Identifiers
  - (Unicode)
- Rules
- Data
- Self-describing Document

Requirements Engineering

- Creation
  - Elicitation
  - Process
  - Analysis
  - Classification
  - Validation
- Use
  - Search/Explore
  - Traceability
  - Requirements Management
  - Design Alternatives
  - Compliance Verification
  - Rationale Management
- Information Synthesis

Logic

Rules

Data

Self-describing Document
Mapping of Requirements (Use Branch) to the Semantic Web Layer Cake

- Requirements Engineering
  - Creation
    - Elaboration
    - Process
      - Analysis
      - Classification
      - Validation
      - Maintenance
  - Use
    - Search/Explore
    - Traceability
      - Requirements Management
      - Design Allocation
      - Compliance Verification
      - Rationale Management
    - Information Synthesis
  - Semantic Web Layers
    - Trust
      - Rules
      - Proof
      - Logic
      - Ontology Support
        - Resource Description Framework + Schema
        - XML - Structured documents
          - Namespaces + XML-Schema
          - Universal Resource Identifiers (Unicode)
Example. Traceability Model for a Metronome

- Compliance Verification Procedures
  - developed_for
  - insured_by
  - verified_by

- Requirements
  - satisfies
  - verified_by

- System Components
  - composed_of
  - derived_from
  - depends_on
  - interface_with

- External Systems
Example. Traceability Viewer Model
XML/RDF Data File for Requirements

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="requirement.xsl"?>

<requirement id="TimeReference.req.xml"
xmlns:="http://www.isr.umd.edu/~selberg"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns="#">

<description>
A physical device which displays a signal at regular frequency.
</description>

<need>
For a performing musician .....  
</need>

<rdf:RDF>
<rdf:Description about="TimeReference.req.xml">

.... lines of rdf code removed ....

</rdf:Description>
</rdf:RDF>

– p. 47/57
Traceability Viewer for a Metronome

XML Requirement Viewer: TimeReference.req.xml

Graphical Navigator

Description
A physical device which displays a signal at regular frequency

Need
For a performing musician it can often be difficult to maintain a constant beat. This is especially true when the music is complex, or very slow and sustained. A physical time reference allows the musician to train his or her internal clock to a higher degree of precision.

Satisfied By
RDF Link: Metronome concept.xml
Basic Observation

- Present-day systems engineering tools organize requirements into tables of requirements and/or hierarchies of requirements.

- These models work well when requirements flowdown and expand from a single source.

But in general, ...

... systems engineering is a team activity. Requirements emanate from multiple sources (some external to the project itself).

This leads to ...

... many-to-many (graph) relationships among requirements.
Visualization of Requirements Hierarchies (Tree View)

Requirements are organized into layers for team development.

Visualization of Requirements in a Tree View.
Example. Tree representation of requirements in SLATE
PaladinRM: Visualization of Requirements

Procedure for Folding a Tree into a Graph

Requirements are organized into layers for team development.

Compaction of the tree representation into a graph.
PaladinRM: Visualization of Requirements

Screendump of Paladin Graphical User Interface (GUI)
A Simple Requirements Graph
Requirements + Properties

PaladinRM: Visualization of Requirements
A Larger Cluster of Requirements

A Requirement graph with requirement clustered into different vertical groups based on their levels
