
PaladinRM: Software for Visualization and Management of System Requirements

Problem Statement

In systems engineering circles, it is well known that requirements management capability improves the likelihood of success in the team-based development of complex multidisciplinary systems. Two key elements of this capability are an ability to identify and manage requirements during the early phases of the system design process. This is when errors are cheapest and easiest to correct. Methodologies for the team development of system-level architectures need to support the following activities:

- Partitioning the design problem into several levels of abstraction and viewpoints suitable for concurrent development by design teams. These teams may be geographically dispersed and mobile.
- Coordinated communication among design teams.
- Integration of the design team efforts into a working system.
- Evaluation mechanisms that provide a designer with a critical feedback on the feasibility of system architecture, and make suggestions for design concept enhancement.

Throughout the development process, teams need to maintain a shared view of the project objectives, and at the same time, focus on specific tasks. It is the responsibility of the systems engineer to gather and integrate subsystems and to ensure ensure that every project engineer is working from a consistent set of project assumptions.

A Typical Application at NASA

Figure 1 is a high-level schematic of layered requirements development at NASA Goddard Space Flight Center. Generally, layers of requirements correspond to levels of responsibility.

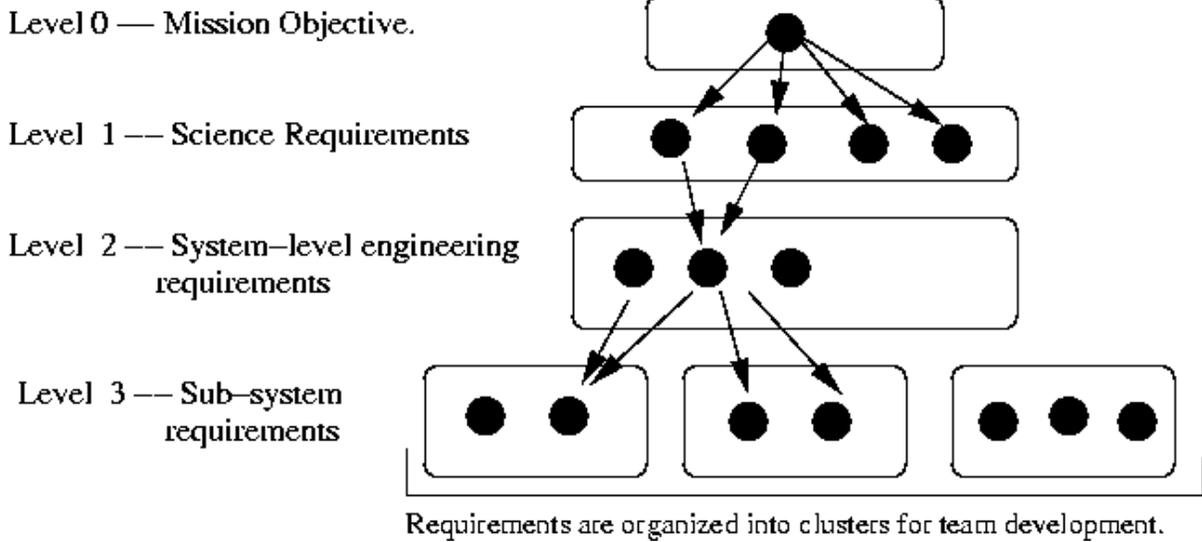


Figure 1. Layered Development and Organization of Requirements at NASA Goddard

Initially, the requirements elicitation process is motivated by a statement of "mission objectives" that will aim to advance our scientific understanding in one or more ways. Levels 1 and 2 focus on the definition of "science" and "high-level engineering" requirements, respectively. At Level 3, engineering requirements are organized into clusters (e.g., ground segment; communications segment; satellite segment) suitable for team development. Requirements at Levels 4 and 5 may be directed to a specific sub-system (e.g., an instrument that will fly on a satellite) or component (e.g., a circuit board).

Representing Requirements with Graph Data Structures

When requirements are organized into layers for team development, graph structures are needed to describe the comply and define relationships among requirements.

As we will soon see, when software tools employ a tree-based model to display relationships among requirements, gaps appear between the visual representation and the underlying graph-based data structures. Systems engineers currently use manual procedures to identify and close these gaps. In an effort to mitigate the limitations of this slow and error prone process, in this study we formulate algorithms and implement software tools for the graph-based organization and visualization of requirements.

State-of-the-Art Capability for Requirements Management

Present-day requirements management tools such as CORE [1], DOORS [2], and SLATE [3] provide the best support for top-down development where the focus is on requirements representation, traceability, allocation of requirements to system abstraction blocks, and recently, step-by-step execution of system models.

State-of-the-art practice is to organize groups of requirements (e.g., functional requirements, interface requirements) into tree hierarchies. Figure 2 shows, for example, a partial requirement document with requirements arranged into layers. Here, requirement C in layer 2 defines requirements E in layer 3. Conversely, requirement E complies with requirement C.

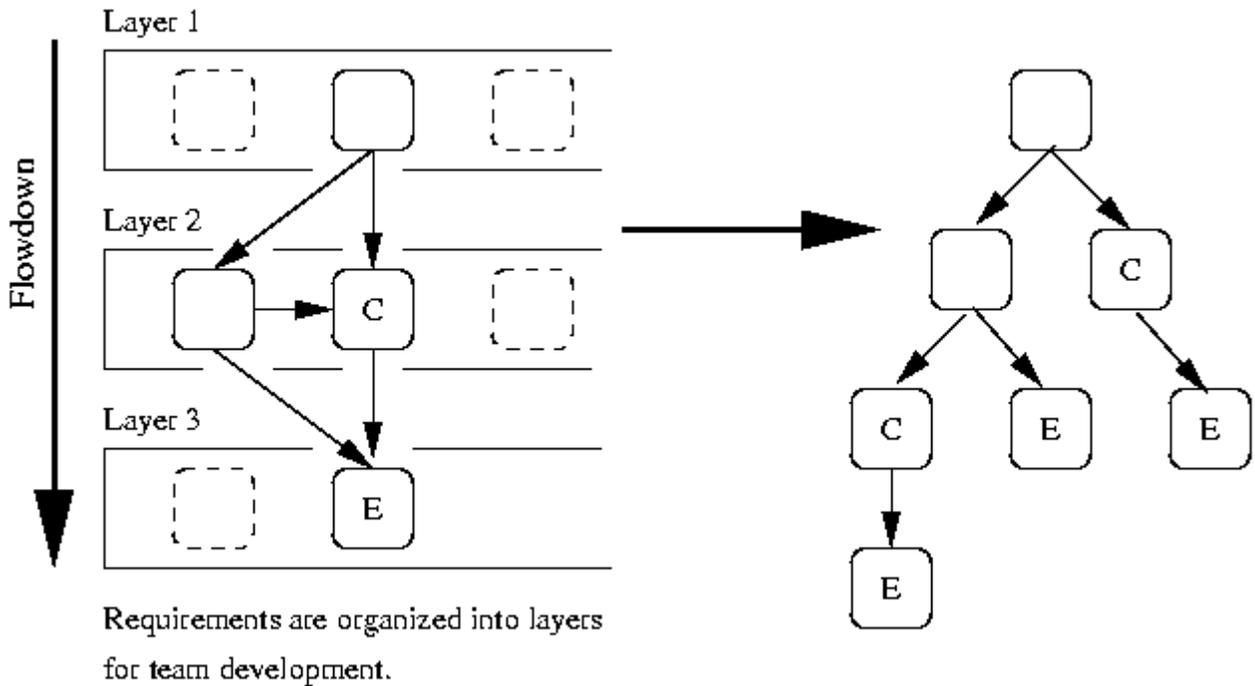


Figure 2. Pathway from Requirements Graphs (organized into layers) to Requirements Tree Hierarchies

A number of complicating relationships are possible. First, a child node (complying requirement) may have more than one parent node (defining requirement). This means that requirements documents need to support multiple inheritance structures. Furthermore, as requirements are classified and broken down into granular components, they can also trace across the same level (i.e., one requirement may comply or define the other requirements).

When the underlying data structure for the requirements is a graph, but visualization procedures assume that a tree structure is sufficient, resolution of this incompatibility is handled through the duplication of nodes in the visual representation (e.g., duplicate entries for nodes C and E in Fig. 2).

Key Problem with Duplicate Requirements

When the overall number of requirements is very small (i.e., less than, let's say, a dozen requirements), the duplication of requirements nodes is not a significant barrier to a complete understanding of relationships in the requirements document. A systems engineer can easily find all of the duplicates and manually reconstruct the graph structure.

However, for all real world problems (hundreds and thousands of requirements), the presence of the duplicate nodes vastly complicates the task of interpreting/understanding the requirements document structure. When the manual identification of all complying and defining requirements becomes intractable, designers cannot be absolutely certain all comply/define relationships have been identified. This leads to decision making based on an incomplete picture of a requirement's role in the system design.

Solution Strategy

The tree representation of requirements hierarchies is fundamentally flawed and, in our opinion, only works well when requirements comply/define from a single source. In this work we formulate algorithms and develop Java-based software that can read the tree representation of the requirement database, and construct and visualize the block diagram representation with all duplicate nodes removed. See Figure 3.

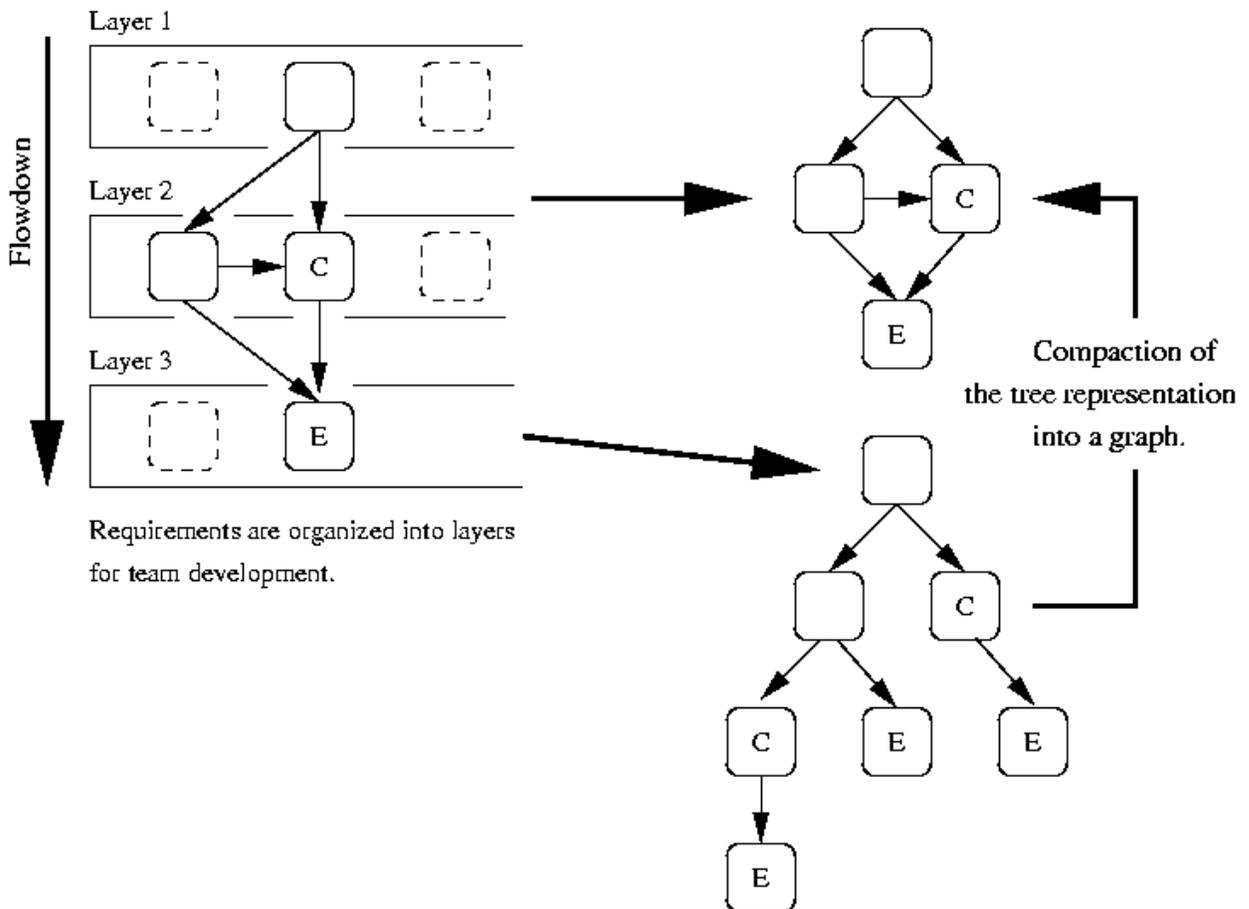


Figure 3. Many-to-Many Relationships in Layers of Requirements. On the right-hand side we show extraction and visualization of requirements as a tree, followed by compaction back in to a graph format.

The input-to-screen transformation involves two steps:

- The tree representation of the requirement document is provided as input and a graph data structure is constructed, which does not have duplicate nodes.
- Construction of a block diagram visualization of this graph structure. This involves using a hierarchical graph layout algorithm to position the requirement nodes on the screen.

Since the heart of this work involves a new and different way of requirements document visualization, efficiency of the systems engineering process can be enhanced with several new features of requirements management and visualization, namely:

- Visualizing a sub-section of the requirement document.
- Differential update of the database of requirements, based on the changes done while visualizing the requirements.
- Annotating the requirement nodes with their attributes of interest.

Screenshots of PaladinRM

We have implemented the requirements visualization software in a tool called PaladinRM -- see Figure 3.

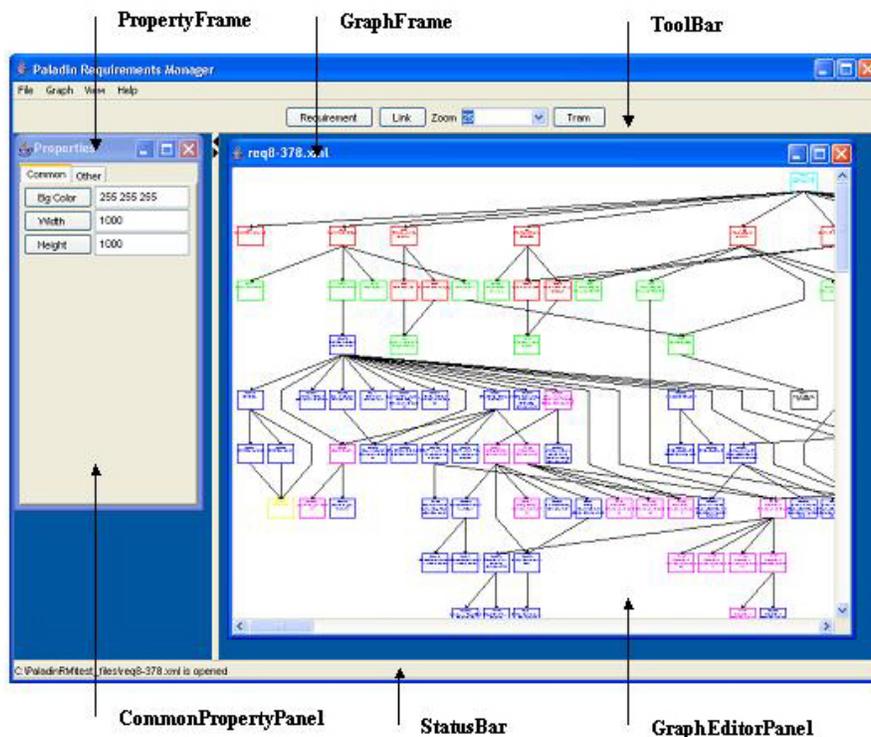


Figure 4. Screenshot of PaladinRM Graphical User Interface

Figure 5 shows a requirement graph with requirements clustered vertically based on their level assignments.

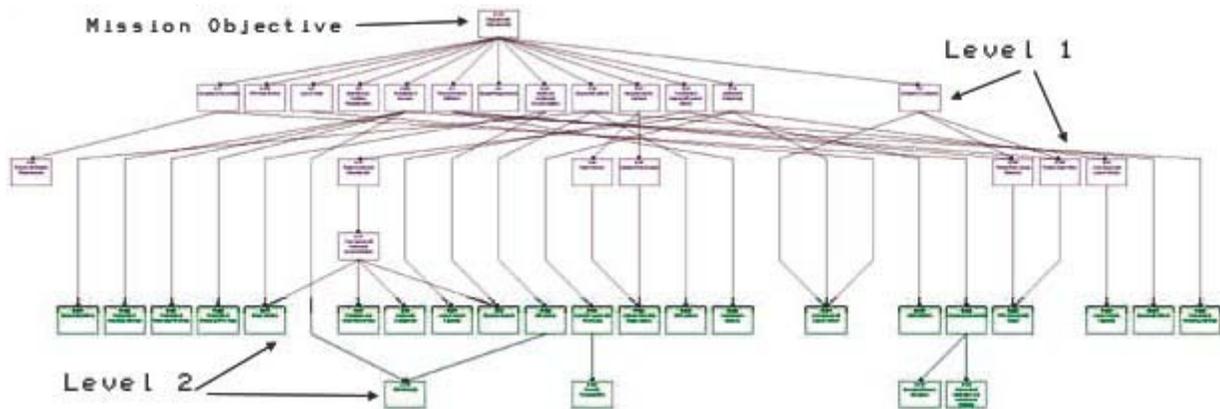


Figure 5. Requirements Graph organized Vertically into Groups (corresponding to layers of development).

PaladinRM has been tested on suites of requirements containing up to 1100 requirement nodes, which currently forms the entire set of GPM project requirements.

Creating Requirements

Requirements are read into PaladinRM through file and interactive keyboard input. Requirements need to be in our proprietary XML file format, which can be created through:

1. The use of a script file that exports requirements from SLATE. (a similar export facility from DOORS is currently in development).
2. Manual development of the xml file with a text editor.
3. Interactive keyboard input. The textual details of requirements can be specified through In the present version, however, interactive keyboard input. attribute details cannot be added through the keyboard.

Development of xml input files is very straight forward and practical for problems having a small number of requirements (e.g., less than 100). [Click here](#) to see a collection of twenty high-level requirements for a building architecture application.

Requirements Selection and Export

PaladinRM allows for the selection of individual requirements followed by a search for requirements that lie in the neighborhood of the selected requirement (i.e., within a user-supplied number of links).

Suppose, for example, that a [small set](#) requirements is read into PaladinRM and visualized as shown in Figure 6.

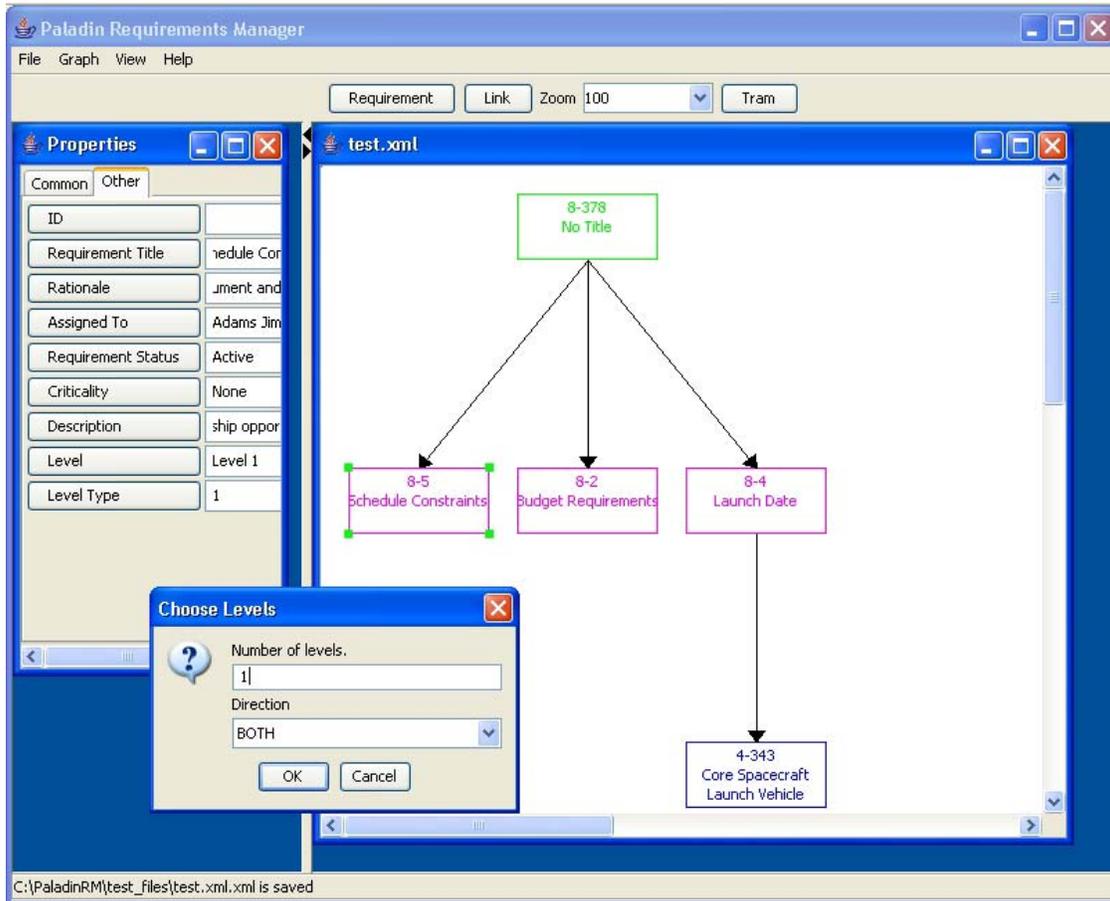


Figure 6. Selection of Requirements within a Neighborhood

Now let's use the selection box (details not shown) to select requirement 8-5 (Schedule Constraint). The properties of requirement 8-5 are shown in the box on the left-hand side of the graphic. A requirements sub-graph is obtained by selecting an individual requirement and then specifying the "number of levels" and "direction" of the required search. In this case, requirement 8-378 (No Title) is the only requirement that is distance one level from 8-5.

The selection box may also be used to select multiple requirements, as shown in Figure 7.

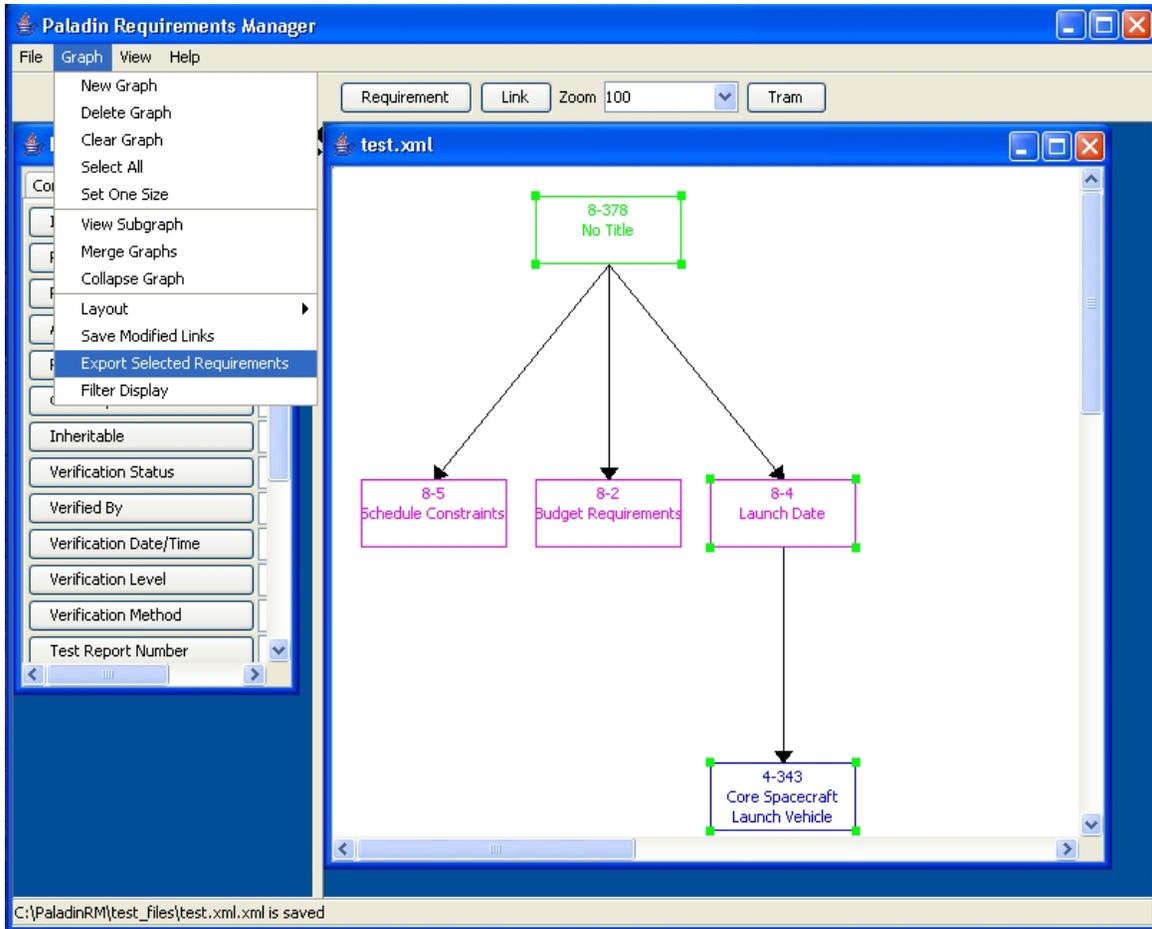


Figure 7. Exporting Selected Requirements

The export command generates an [xml file](#) of requirements that have been selected.

Future Work

Support for graph-based visualization and management of requirements is simply a first step in systems development.

Looking ahead, we would like to understand how graphs of requirements can be attached to class and object hierarchies (representing concepts and objects in a particular application domain), which in turn might be mapped onto engineering drawings.

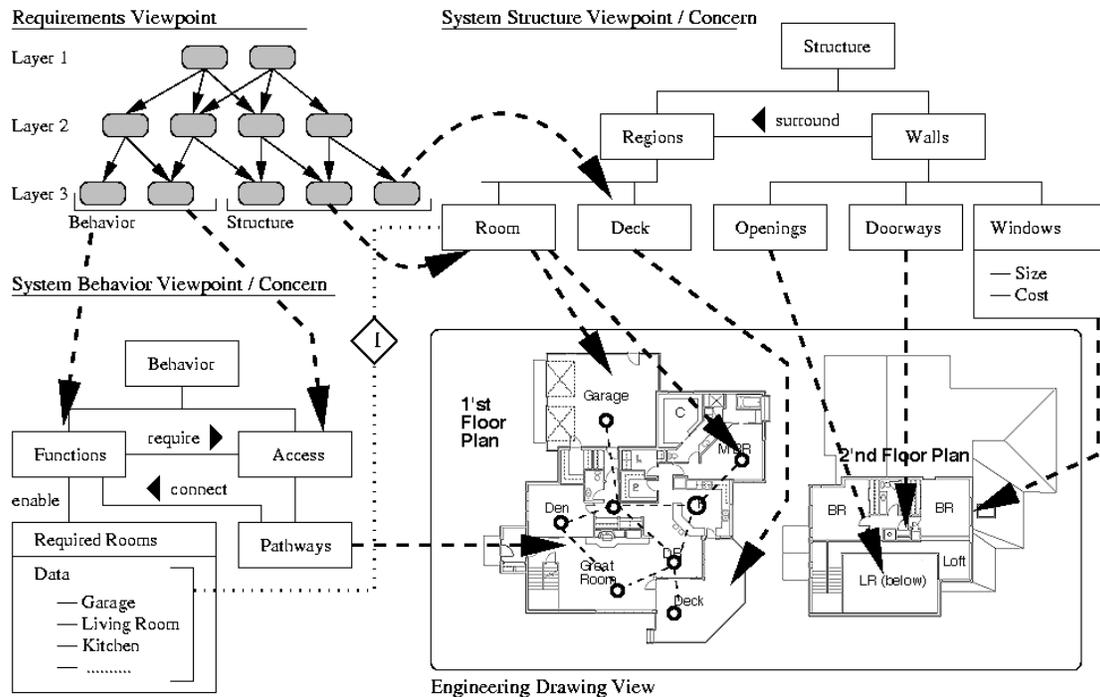


Figure 8. Pathway from requirements to UML-like class diagrams to engineering models/drawings for an architectural design application.

Figure 8 is a schematic illustrating the flow of requirements to UML-like representations of system behavior and system structure, which in turn, flow to engineering models/drawings.

Acknowledgements

This work was supported, in part, by a series of grants from NASA Goddard Space Flight Center. The views expressed in this web page are those of the authors and not necessarily those of the sponsors.

Software Distribution

To license the PaladinRM software, please contact

Jim Poulos,
 Office of Technology Commercialization,
 University of Maryland,
 College Park,
 Ph. (301)-403-2711

E-mail: jpoulos@umd.edu

References

1. CORE, 2003. See <http://www.vitechcorp.com/productline.html>
 2. Dynamic Object Oriented Requirements System (DOORS), 2003. See <http://www.telelogic.com/products/doorsers/doors/>
 3. System Level Automation Tool for Enterprises (SLATE), 2004. See <http://www.sdrc.com/slate/>
-

Developed in December 2004 by Mark Austin
Copyright © 2004, Vimal Mayank, Natasha Shmunis (nee Kositsyna), Mark Austin,
University of Maryland. All rights reserved.

XML Code for Requirements

This input file contains xml markup for five high-level requirements in the NASA GPM project. The requirements are organized into three layers.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--comment SLATE Database sampled 11/26/2003 11:20:25 -->
<ReqFlow>

<Req ROIN="8-378" type="Mission Objective" level="0">
  <Title text="Programmatic Requirements"/>
  <Description text="The GPM shall adhere to program
    requirements as agreed between the NASA Earth Science
    Enterprise and the NASA/GSFC GPM program office."/>
  <ReqList>
    <CompReq ROIN="8-2"/>
    <CompReq ROIN="8-4"/>
    <CompReq ROIN="8-5"/>
  </ReqList>
  <Attribute text="Assigned To" value="None"/>
  <Attribute text="Change Proposals Allowed" value="Not Assigned"/>
  <Attribute text="Criticality" value="None"/>
  <Attribute text="ID" value=""/>
</Req>

<Req ROIN="8-2" type="Level 1" level="1">
  <Title text="Budget Requirements"/>
  <Description text="To Be Determined"/>
  <Attribute text="ID" value=""/>
  <Attribute text="Requirement Title" value="Budget Requirements"/>
  <Attribute text="Rationale" value="Imposed by HQ."/>
  <Attribute text="Assigned To" value="Adams Jim"/>
  <Attribute text="Requirement Status" value="Active"/>
</Req>

<Req ROIN="8-4" type="Level 1" level="1">
  <Title text="Launch Date"/>
  <Description text="The launch of the GPM Core spacecraft
    shall be planned for within the US
    Government Fiscal Year 2008. "/>
  <ReqList>
    <CompReq ROIN="4-343"/>
  </ReqList>
  <Attribute text="ID" value=""/>
  <Attribute text="Requirement Title" value="Launch Date"/>
  <Attribute text="Rationale" value="This requirement is based on
    the JAXA development schedule as well as a reasonable
    development time and funding profile for the NASA elements."/>
  <Attribute text="Assigned To" value="Adams Jim"/>
  <Attribute text="Requirement Status" value="Active"/>
  <Attribute text="Criticality" value="None"/>
  <Attribute text="Inheritable" value="Yes"/>
  <Attribute text="Verification Status" value="None"/>
  <Attribute text="Verified By" value="None"/>
  <Attribute text="Verification Date/Time" value="YY/MM/DD-24:00"/>
  <Attribute text="Verification Level" value="Segment"/>
  <Attribute text="Verification Method" value="Analysis"/>
  <Attribute text="Test Report Number" value=""/>
  <Attribute text="Qualification Date/Time" value="YY/MM/DD-24:00"/>
  <Attribute text="Requirement State" value="Uncontrolled"/>
  <Attribute text="Change Proposals Allowed" value="Not Assigned"/>
  <Attribute text="Verification Description" value=""/>
</Req>
```

```

<Req ROIN="4-343" type="Level 2" level="2">
  <Title text="Core Spacecraft Launch Vehicle"/>
  <Description text="The Core Spacecraft shall be compatible for
    a flight in early 2009 on the H-IIA 202 with a 1194M payload
    adapter in a 4/4D-LS fairing."/>
  <Attribute text="ID" value=""/>
  <Attribute text="Requirement Title" value="Core Spacecraft Launch Vehicle"/>
  <Attribute text="Rationale" value="JAXA is providing a launch
    vehicle to launch GCOM-A1 and GPM-Core together from TnSC, Japan."/>
  <Attribute text="Assigned To" value="None"/>
</Req>
<Req ROIN="8-5" type="Level 1" level="1">
  <Title text="Schedule Constraints"/>
  <Description text="The GPM development shall be synchronized with
    the JAXA instrument and launch vehicle schedules to maximize
    partnership opportunities. Also, where appropriate, programmatic
    schedules will be synchronized to enhance other international
    partnership opportunities."/>
  <Attribute text="ID" value=""/>
  <Attribute text="Requirement Title" value="Schedule Constraints"/>
  <Attribute text="Rationale" value="To realize the full impact of GPM,
    foreign partnerships are required, so the GPM schedule must align
    with the schedules of those partners. The JAXA partnership has the
    highest priority since the primary spacecrat depends on this
    partnership for its instrument and launch."/>
  <Attribute text="Assigned To" value="Adams Jim"/>
  <Attribute text="Requirement Status" value="Active"/>
  <Attribute text="Criticality" value="None"/>
</Req>
</ReqFlow >

```