Requirements Decomposition Analysis

Dr. Martin S. Feather, Dr. Allen P. Nikora

Jet Propulsion Laboratory,
California Institute of Technology

&

Dr. Jane Oh

Wayne State University
(JPL Summer Faculty Fellow, 2003)

Presented by Martin.S.Feather@Jpl.Nasa.Gov

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Work sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA Software IV&V Facility. This activity is managed locally at JPL through the Assurance and Technology Program Office.

Roadmap

- Motivation – Mars Polar Lander
  Example of flawed requirements decomposition
- Requirements Decomposition:
  Means to perform analysis (check decomposition done correctly)
- Example 1: Study of Science Experiment's Requirements
  - Simple consistency checking
  - Additional complexity arises when timelines involved
- Example 2: Study of an Entire Flight Project's Requirements
  - Sheer volume of requirements!
  - *Existing* capabilities for traceability
  - *New* capability for traceability closure
  - Extracting relevant requirements *without* relying on existing traceability
    - Motivation
    - *New* capability for text-search based retrieval, & example
- Status & Future

JPL Motivation – Mars Polar Lander

3.7.2.2.4.2 *Wait a minute, how many requirements were there?*

SYSTEM REQUIREMENTS

1) The touchdown sensors shall be sampled at 100-Hz rate.

The sampling process shall be initiated prior to lander entry to keep processor demand constant.

However, the use of the touchdown sensor data shall not begin until 12 meters above the surface. X

2) Each of the 3 touchdown sensors shall be tested automatically and independently prior to use of the touchdown sensor data in the onboard logic.

The test shall consist of two (2) sequential sensor readings showing the expected sensor status.

If a sensor appears failed, it shall not be considered in the descent engine termination decision.

3) Touchdown determination shall be based on two sequential reads of a single sensor indicating touchdown.

FLIGHT SOFTWARE REQUIREMENTS

3.7.2.2.4.2

Processing

a. The lander flight software shall cyclically check the state of each of the three touchdown sensors (one per leg) at 100 Hz during EDL.

b. The lander flight software shall be able to cyclically check the touchdown event state with or without touchdown event generation enabled.

c. Upon enabling touchdown event generation, the lander flight software shall attempt to detect failed sensors by marking the sensor as bad when the sensor indicates "touchdown state" on two consecutive reads.

d. The lander flight software shall generate the landing event based on two consecutive reads indicating touchdown from any one of the "good" touchdown sensors.

Requirement didn't flow down

Mars Polar Lander / Deep Space 2 Loss – JPL Special Review Board Report

JPL D-18709 – page 120

Requirements Decomposition ⬇

Level1 Requirements ⬇ System Requirements

Level2 Requirements ⬇ Software Requirements

Level3 Requirements ⬇ *a.k.a. "Requirements Flowdown"*

…

Analysis

If decomposition is incorrect between a pair of levels' requirements, how might this be detected?

- **Knowledgeable designers/coders/testers notice something amiss**
  Not thorough or systematic (CMM level 1)
- **System fails validation test (behavior does not match high-level requirements) Imperfect – can't test everything**
- **Inspection/review of the two levels of requirements**

This CI's purpose is to improve our ability to do this!

JPL Study: Science Experiment's Rqmts

(identities are being deliberately obscured in this material)

α

3.1.6 α shall respond to engineering event notification (via γ DB update) and generate an appropriate response mission replan (sent to γ) within 24MB RAM (peak memory size) and 30625 MI, execution of this plan shall result in less than 0.2% (= 2% α * 10% γ) frequency invalid commands (again sent from <System> to the … FSW, and as determined by flight safety rules) with less than 0.02% frequency invalid commands (from <System> to the … FSW) (expt 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, <T>: cpu 2.1.6, ram 2.1.7)

β

2.1.7 The β CDS shall have 49.5 MB of RAM per spacecraft continuously available for the <System> software image to run during <System> experiment time. Of this only 24MBytes is for <System>-specific planning software; the remainder is carried in the CM RAM budget.

Running Code Size

γ Core + scripts/rules	0.5MB
γ Database	3MB
α	24MB
Observation Planner	2MB
Cluster Manager Logging	20MB
Total	**49.5MB**

Simple consistency checking

JPL Study: Science Experiment's Rqmts

β 2.2.2 β shall provide ability to <…>, 512K bytes every 3 days during <System> allocated time, (uplink 50k bits per second) (est. 600s per uplink) (est. 3 contacts per day) (for 11.25Mbits mission uplink per day) Uplink data to be required no more than two days before uplink.

Timelines introduce additional complexity

Level 2 and level 3 requirements: approximately 9 pages in total

JPL Study 2: An Entire Flight Project

(identities being deliberately obscured in this material)

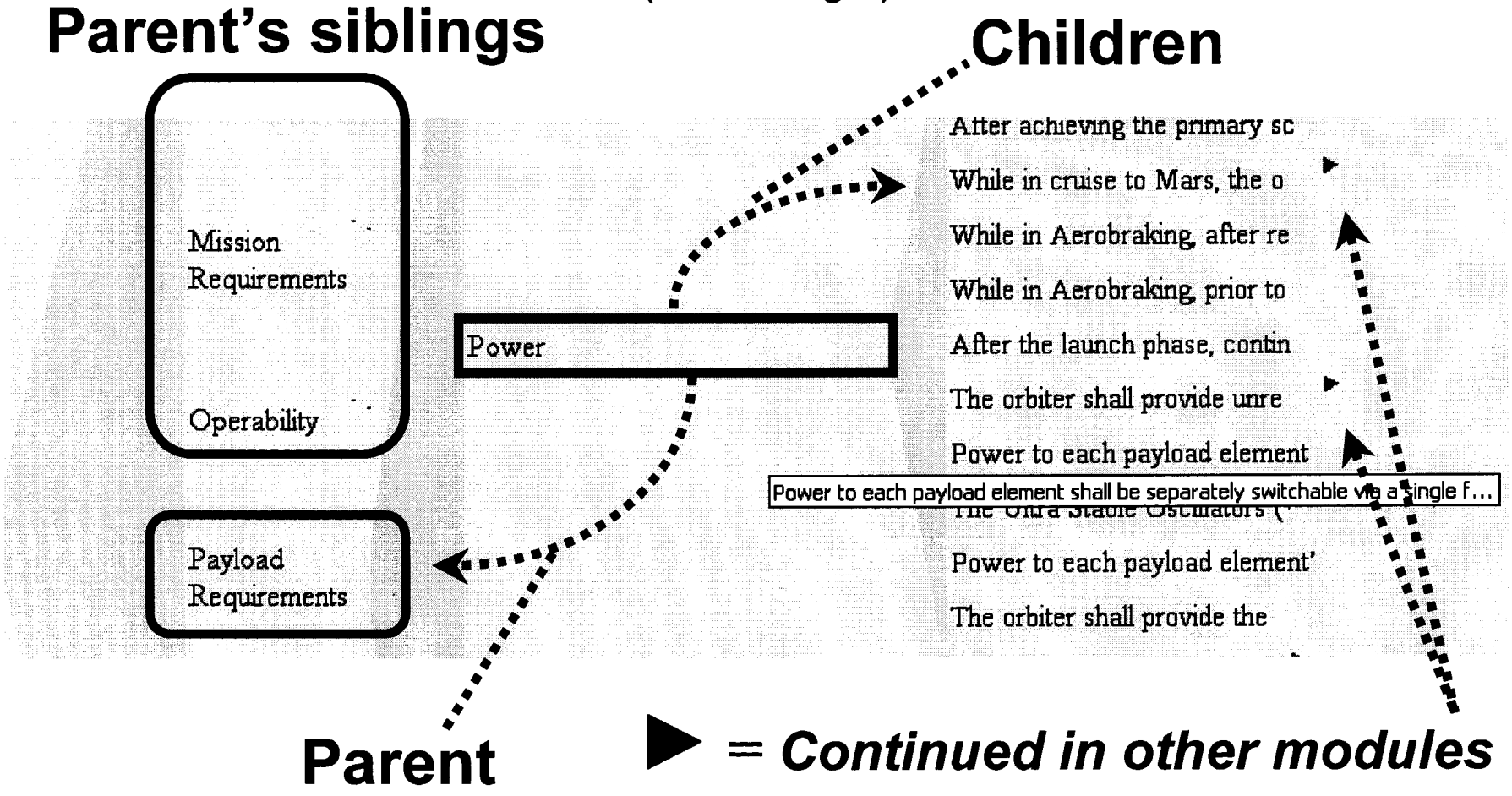
- (1 of) Spacecraft requirements module
  660 requirements
- (1 of) Mission operations requirements module
  235 requirements
- (5 of) Level 2 requirements modules:
  1,370 requirements
- (1 of) Level 2.5 requirements module
  550 requirements
- (5 of) Level 3 requirements modules:
  840 requirements
- (8 of) Level 4 requirements modules:
  1,595 requirements

This project (and others) keeps Requirements & traceability information DOORS © Telelogic

Scale: sheer volume of requirements!

JPL Tracing Decomposition Structure *Existing* Capabilities

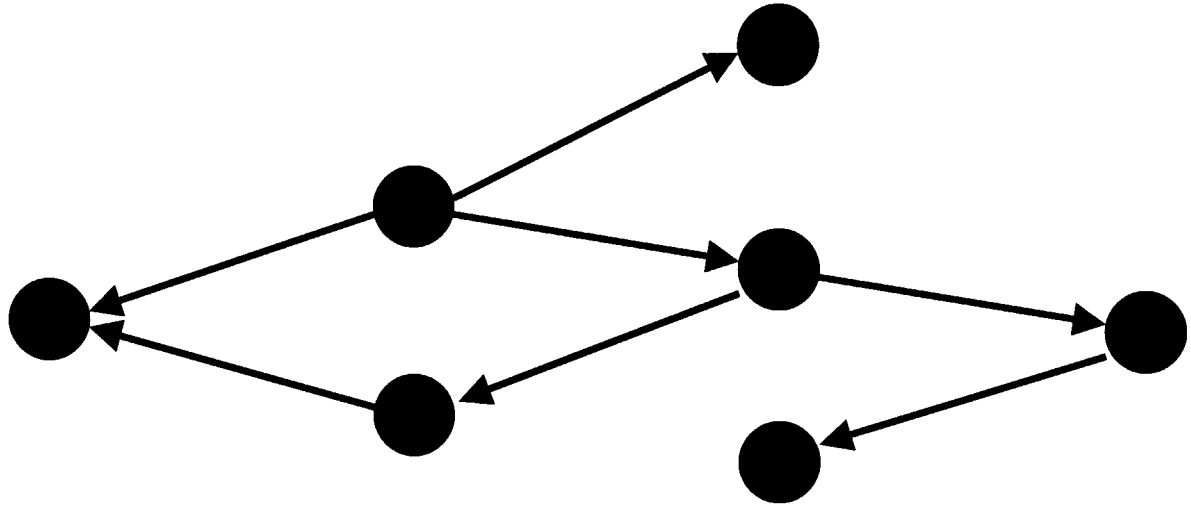
Partial screenshot of DOORS in "graphics mode"
(© Telelogic)



JPL Tracing Decomposition Structure *New* Capability

Transitive closure of traceability

Surprise (to me) – islands of connectivity (i.e., closure ≠ all requirements)



- Automated script
- Traverses traceability links in *both* directions
- Traversal continues over module (level) boundaries
- Results assembled into single hyperlinked HTML table
- Yields comprehension of decomposition neighborhood (e.g., if considering impact of changing a requirement)
- Future refinements possible (e.g., inhibit tracing through…)

Why?

While the Project, SQA & IV&V should and will continue to use existing traceability information (e.g., does every requirement trace to a test case? Do the traced-to requirements achieve the traced-from requirement?), nevertheless…

For assurance purposes, don't want to *rely* on completeness and correctness of existing traceability information.

JPL Extracting Relevant Requirements *Without* using Existing Traceability

From the ENTIRE set of project requirements, extract only those dealing with a resource, e.g., mass allocation

Search for "mass" (potentially many results)

Search for *units* of mass: "kg" (potentially many results)

Search for " [digit] kg " (yields small set of results)

Note: may need to search for multiple units, e.g., bits per second expressed in Gbps, Mbps, Kbps and bps!

[0-9]( )[G|g|K|k|M|m|()]bps

JPL Extracting Relevant Requirements *Without* using Existing Traceability

"[0-9]( )[K|k]g" yields 15 requirements dealing with mass.
Here are several of them:

Spacecraft Requirements:

The spacecraft shall be designed for a maximum orbiter launch mass of 2180 kg and shall accommodate the following allocations:

Spacecraft Wet Mass - 1988.0 Kg
Instruments - 139.0 Kg
+
E… - 17.5 Kg
Project Manager Reserve - 35.5 Kg

Project System Requirements:

Launch Mass: The PS shall be designed for a maximum injected mass of 2180 kg.

The PS shall be designed to accommodate 156.5 kg [including all reserves] of Payload mass.

The PS shall accommodate a reserve of 35.5 kg to be allocated by the project manager

Status & Future

- ACCOMPLISHED
  - Automated transitive closure script:
    - implemented as PERL script that operates on HTML files as exported from DOORS
  - Regular-expression text-based searches:
    - carried out using DOORS search and filter capabilities:
      - search the project to identify the modules
      - filter those modules to just those requirements
- FUTURE WORK:
  - Implementation:
    - Both could be implemented as scripts within DOORS (using DOORS scripting language DXL)
    - Both amenable to other requirements repositories
  - *Refine closure script! Extend searches! Look at checking!*