

Design Exploration for Spacecraft Flight Software

Dr. Christopher Landauer Computers and Software Division The Aerospace Corporation

2011 Spacecraft Flight Software Workshop JHU/APL 19-21 October 2011

© The Aerospace Corporation 2009

Outline

Background and Context

- Context: Space Systems
- Software Caused Failures
- U.S. Government Accountability Office

Overview of Approach Conclusions and Prospects Preview Details of Approach

- Wide Software Trade Space
- Optimization and Analysis Methods
- Visualization and Exploration Methods

What We Get if All of this Works Example Problem Description and Setup Conclusions and Prospects Recap

Context: Space Systems

Background: Space systems are the most complex systems humans build that work

- Hundreds of organizations, thousands of people, millions of lines of code
 - . Ground facilities and satellites, manufacturing, integration and operation
- Developed over (and usually expected to last for) decades

These systems are too complicated to rely on human management

- These systems must largely manage themselves
- ALL systems are autonomous systems
 - . The space segments of space systems are especially expensive to repair

Software has become our "intellectual caulking material"

- Manages interactions among less flexible components
 - . Hardware components and execution environments
- Increasing expectations has meant increasing complexity of software
- We are already past the stage of complexity overload
- We need adequate principled methods, much more than "heroic engineering"

Software cannot be developed in isolation

- ALL systems are embedded systems

We need system engineering of software, not just software engineering

Software Caused Failures

The first Ariane 5 launch (04 June 1996)

- ESA expected no remaining errors in the Ariane 4 software
- After dozens of successful launches.

Major military satellite launch (April 1999)

- Two people used different conventions for scientific notation.

NASA's Mars Climate Orbiter mission (23 September 1999)

- Different developers used different physical units in thruster commands.

All three have something in common

- Attributable to insufficiently careful software or software design
- Something quite simple went wrong

The problem is that there are many millions of simple things that can go wrong.

We have no development methods capable of managing this complexity.

U.S. Government Accountability Office

Since 2006, the GAO has repeatedly identified "reliance on immature technologies" as a key factor in delays and cost over-runs of recent military space programs - And specifically mentioned software as a cause of delays

They suggested that the Air Force and their contractors not "reach beyond their grasp".

The author fundamentally disagrees with this suggestion

The author believes that military space must reach as far as possible beyond the recognized state-of-the-art in all technical areas.

- Familiar system development techniques clearly cannot be used for that.

The rapid rise in performance of space-qualified computing systems

- Makes many old system engineering design decisions no longer justified
- And in many cases incorrect.

We need to turn towards first principles in space software architectures

Overview of Approach

Our approach is to return as much as necessary to first principles, and consider:

- Mission instruments (usually assigned to the payload)
- Mission support functions
- Fault management (including safehold)
- Software requirements for all of these
 - . Data processing and movement requirements
 - . Reliability, availability, and timeliness requirements

The approach presumes:

- Multiple resolution / capability / performance levels for every component of software
 - . Supports graceful degradation
- Flexible assignment of processes to processors
- Parallel model-based software / hardware development
 - . Software development and testing can begin before hardware is constructed

Enabling concepts and tools that we need to provide include

- Software configuration space identification
- Architecture evaluation methods
- Hardware modeling and analysis tools

Conclusions and Prospects Preview

This approach avoids some of the assumptions that traditional design practice makes

- A priori separation of spacecraft from payload processing
- Fixed assignment of processes to processors
- Fixed processor architecture with redundancy
- Relatively few large processors

We think it can also avoid some of their consequences

- Complicated processor sharing and scheduling methods
- Many failures cause abrupt and permanent mission degradation

We are exploring various technical aspects of such a system

- Differential geometry shapes of complicated surfaces
- Algebraic geometry solutions of constraint equations
- Optimization and exploration algorithms for high-dimensional constraint spaces

We are trying to imagine others

- Preferential geometry is about how things change using "importance"
- Driver identification is about which variables can be ignored and for how long

We think this approach can advance the state of the art in flight software design

Start with a Very Wide Software Trade Space

Aspects of a space architecture managed by software include

Mission

- Surveillance systems have sensors that collect data (e.g.) Mission support

- Data processing and movement

- Communications, telemetry, and commanding Hardware management

- Orbit, attitude, and configuration control

- Thermal and electrical control

Software management

- Resource selection and application

- Monitoring with "sanity checks" Fault management

- Behavioral consistency observation

- Detection, diagnosis, mitigation Safehold

- Guaranteeable protective behavior

Aspects of Software Trade Space

Each software architecture runs on a hardware configuration

Each hardware configuration is defined by

- Sets of processors and data links
 - . Processing and transmission capabilities
 - . Reliability and susceptibility ratings
- Connections among the computers and busses

Each software architecture is defined by

- Sets of data processing and movement requirements
 - . Volume and timing (latency and throughput)
 - . Dependencies among processes and communications
- Reliability, availability, and recovery requirements
- Parallelization opportunities

Each architecture configuration is defined by the assignment of

- Processes to processors
- Communications to links

Optimization and Analysis Methods

Problem definition

- We take some of the technical requirements as constraints

- . Some constraints are derived from physics and will not change
- . Some are derived from mission class and will not change
- We take some of them as subject to improvement or weakening . Soft constraints and opportunities

Multiple-Objective Optimization in Constrained Spaces

- Objectives include reliability, cost, performance
 - . Cost is presumed related to

Number, power, and reliability of processors Number, throughput, and reliability of busses

Some variables are qualitative or discrete

- Preference assignments (some objectives are more important)
- Parallelization assignments of dependency graphs
 - . Separation into independent processing threads
- Number of processors

Optimization and Analysis Methods (Continued)

These are local constraint spaces

- There may be different constraints in different places (ref 1993 VEHICLES paper)
- Different variables are more important in different places

Once we have a solution to the optimization problem

- We need to examine our assumptions
- Any choices we have made to specialize the optimization must be re-examined . Sensitivity analysis (ref 1990 sensitivity paper)

We are interested in the robustness of solutions

- Not "best", but "good in a big region of good"
- We therefore want to explore the surroundings of a point solution

Another approach is to use guided negotiation among local optimizers

- Each optimizer is responsible for one performance variable
- Guidance is by human experts
- Allows explicit study of trade-offs

Visualization and Exploration Methods

Your intuition in high n-dimensional space is wrong

- Hypervolume ratio of n-sphere/n-cube --> 0 faster than (n/e) ^ -n
- Hypervolume ratio of (10% n-annulus)/n-sphere --> 1 as 0.9 ^ n



- Almost all of the space is near the boundaries

We therefore need to reduce the dimension to promote understanding

- Preferential geometry chooses some dimensions as more important than others
- Combine several dimensions using an amalgamation function . Need to study the effects of the choice of function
- Select a few local variables and study their interactions

Exploration in n-dimensional constrained spaces (the "Grand Tour")

- Multi-dimensional grid displays

What We Get if All of this Works

Process for mapping program expectations into processing requirements - Independent of hardware architecture

Process for determining solution spaces that satisfy those requirements - With sensitivity information about the effects of each requirement

Methods for analyzing, exploring, and visualizing those solution spaces

Sets of appropriate software architectures

- With reliability and availability measurements

Sets of appropriate processor and bus architectures

- With assignment of processes to processors
- With software migration paths for failure response

New methods for evaluating new kinds of spacecraft processing architectures

Example Problem Description

The example mission is earth surveillance

The sensor collects circular frames 10 times every second

- Radius 4096 pixels, so pi * (2 ^ 12) ^ 2 ~ 57.7M pixels per frame
- Initial on-board processing reduces it to up to 1000 detection reports per frame
 - . Suppression of clutter and identification of potential points of interest
 - . Defined by 5x5 pixel squares, which gives up to 25K pixels per frame
- Subsequent ground processing identifies certain patterns of interest
 - . There are different ranges in which patterns may be detected
 - . Detection reports in different ranges may be the same phenomenon
 - . Different phenomena of interest are mostly independent

It is essential for this data processing to know

- Where the spacecraft is
- What direction the sensor is pointed
- What detection range the sensor is using

It is not required that the spacecraft move the sensor or the spacecraft

- Orbit is relatively stable, sensor is fixed to the spacecraft
- So attitude stability is important

. Though very slow rotation could reduce sensor fatigue

Example Problem Setup

Data movement of 57.7M pixels (= 462M bytes) from sensors to processors

- Each pixel is 8 bytes (for the 8 different pattern ranges selected)
- Every 0.1 second
- Parallel data paths are expected (and likely necessary)

Data processing from 57.7M to 25K pixels (= 200K bytes) per frame

- Accumulation of individual sensor local calibration data

. Separately for each sensor (4 bytes per pixel)

- Parallel computation is expected (and likely necessary)

Transmission of processed data to ground

- Meta-knowledge (8 bytes per pixel) for up to 1000 reports per frame
 - . Meta-knowledge describes coordinates of report pixels and calibration data
- Total is about 4M bytes per second
- Calibration data is interspersed when there aren't many reports

Mission support operations

- Monitor sensor and data path health
- Monitor processor health

Conclusions and Prospects Recap

This approach avoids some of the assumptions that traditional design practice makes

- A priori separation of spacecraft from payload processing
- Fixed assignment of processes to processors
- Fixed processor architecture with redundancy
- Relatively few large processors

We think it can also avoid some of their consequences

- Complicated processor sharing and scheduling methods
- Many failures cause abrupt and permanent mission degradation

We are exploring various technical aspects of such a system

- Differential geometry shapes of complicated surfaces
- Algebraic geometry solutions of constraint equations
- Optimization and exploration algorithms for high-dimensional constraint spaces

We are trying to imagine others

- Preferential geometry is about how things change using importance
- Driver identification is about which variables can be ignored and for how long

We think this approach can advance the state of the art in flight software design

Selected References

These are the references cited in the presentation.

Christopher Landauer, Grace Chen-Ellis,
"Sensitivity Analysis for Conceptual Design",
pp. 410-414 in
Proceedings of Interface'90: The 22nd Symposium on the Interface (between Computer Science and Statistics),
17-19 May 1990, East Lansing, Michigan (1990)

Kirstie L. Bellman, April Gillam, Christopher Landauer,

"Challenges for Conceptual Design Environments: The VEHICLES Experience", Revue Internationale de CFAO et d'Infographie}, Hermes, Paris (September 1993)