

Independent Assessment of Two NASA Fault Management Software Architectures

Phillip Schmidt
Donald Brueck
Mark Rokey
Joseph Pope

The Aerospace Corporation

2012 NASA Fault Management Workshop
April 10-12, 2012, New Orleans, LA

PUBLIC RELEASE IS NOT AUTHORIZED. Distribution limited to NASA and their US Contractors only. Further dissemination only with the approval of NASA.

EXPORT WARNING. Portions of this presentation may contain export controlled technical data within the definition of the *International Traffic in Arms Regulations* (ITAR) and therefore subject to the export control laws of the US Government. Transfer of this data by any means to foreign persons or their representatives, whether in the US or abroad, without a validated export authorization or approval from the US Department of State, is prohibited. Violations of ITAR are subject to fines, imprisonment, or both, under Title 22, United States Code (U.S.C.), Section 2778, Control of Exports and Imports and title 50 U.S.C., Appendix 2410, Violations.

DESTRUCTION NOTICE. When this document is no longer required, destroy by any method that will prevent reconstruction of the information.

Agenda

- Background
- Assessment Criteria
- FM Architecture Overview
- Significant Findings
- Observations
- Recommendations



Background - Objectives

- Marshall Space Flight Center (Discovery/New Frontiers Program Office) requested The Aerospace Corporation to perform an independent, objective assessment of the two fault management (FM) software architectures
 - *Architecture A: Layered, collaborative based FM*
 - *Architecture B: Event driven, rule-based FM*
- Review project materials from different NASA programs for each architecture
- Conduct interviews with subject matter experts for each architecture
- Assess FM software architecture
- Report significant findings
- Provide observations/recommendations



Background - Ground Rules

- Study focused on fault management software architecture impacts for deep space robotic missions
- Programs using architectures had different risk profiles, cost/schedule constraints
- NASA deep space missions have unique characteristics/trades
 - *Low bandwidth, high latency vs. high bandwidth, lower latency*
 - *Single vs. full redundant configurations*
 - *Adaptiveness/complexity vs. predictability/simplicity*
 - *Autonomous vs. ground control*
 - *Fail operational (maintain mission) vs. fail safety (spacecraft safety/payload shedding)*
- Primarily a qualitative assessment



Assessment Criteria and Quality Areas

- Criteria were categorized according to a project's timeline
 - *Planning*
 - *Requirements*
 - *Design*
 - *Implementation*
 - *Test and verification*
 - *Operations*
- Criteria were used in reviews and fact-finding interviews
- Criteria relate to 2008 NASA FM Workshop recommendations (details in backup)



Specific FM Architectural Assessment Criteria

- **Planning**
 - *Clarifies organization of resources and roles*
 - *Balances flexibility and complexity in early design choices and test effort impacts*
- **Requirements**
 - *Addresses program unique mission requirements to provide a reliable, safe, FM capability*
 - *Considers the entirety of FM functionality when allocating requirements to HW, SW and operations*
- **Design**
 - *Is well coordinated and appropriate to mission needs*
 - *Utilizes architectural representation and analysis techniques that improve the design, implementation and mitigate risk*
 - *Manages system complexity of robotic deep space missions and considers spacecraft robustness early in the design*



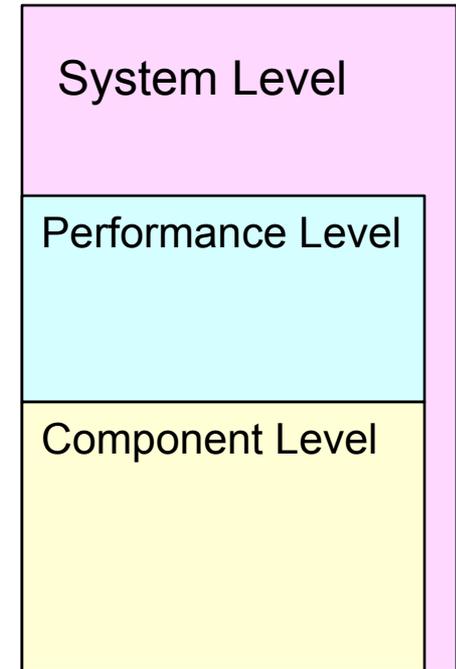
Specific FM Architectural Assessment Criteria

- **Implementation**
 - *Facilitates reuse*
 - *Easily adapts to change*
 - *Demonstrates reliable FM capabilities/features for specified mission*
- **Test and Verification**
 - *Facilitates test design and execution*
 - *Identifies the level of risk exposure*
- **Operations**
 - *Provides understanding of operational impacts on design and implementation*
 - *Facilitates ease of operations*



FM Overview: Architecture A

- Object-oriented, collaborative logic approach
 - *FM decision logic distributed across objects in flight software*
- Layered control design
 - *Component* - objects interfacing with HW but provide virtualized component to detect and respond
 - *Performance* - objects monitoring subsystem domain areas
 - *System* - objects performing S/C level functions
- Each layer
 - *Relies on lower layers to perform more primitive functions*
 - *Provides services to higher layers*
 - Uses a “pull” approach to provide fault data as needed to upper layer
 - *Responses not allowed to affect other objects at same or higher level but can manipulate lower level object*



Architecture A adapts well when virtual components can be reused



FM Overview: Architecture B

- Centralized, rule-based, monitor-response FM
 - *Rules provide fault monitoring if-then conditions to handle anomalies*
 - *Triggered rules invoke a macro response*
 - *Macros contain stored command sequences to address fault*
 - *All rules execute at 1Hz rate*
- FM decision logic is not compiled into the flight software
 - *Multiple rules and macros are interpreted at run-time*
 - *Rules are prioritized so response macros can be pre-empted before completing*

Architecture B provides adaptive rule/macro interpretation

This chart has not been approved for public release and is subject to the restrictions on the title chart of this material.



Significant Findings: Planning and Requirements

- **Planning improved on later projects**
 - *Architecture familiarity/maturity*
 - *Strong reliance on prior human-directed program experience*
 - *Identifying differences in current program relative to heritage was significant to avoid cost/schedule surprises*
- **Formalized methodology to analyze FM mission behavior/interactions was lacking**
 - *Strong reliance on engineering judgment*
 - *No formalized use case scenario development (e.g. off-nominal cases)*
 - *No formalized analysis of state variable interactions, fault coverage, contention/race conditions*
 - *SW FMEA (Failure Mode and Effects Analysis) was not practiced to identify SW risk mitigations and/or SW requirements*
- **Some processes risked late identification of requirements**
 - *Reliance on later testbed environments vs. early sim modeling/prototypes*
 - *Many lower level requirements are the result of FMECAs (Failure Modes Effects and Criticality Analysis) and fault trees that may not be available until late in the program raising risk for late requirements*

Hero-driven engineering judgment will not scale with mission complexity



Significant Findings: Design and Implementation

- Common strengths
 - *Tailoring multiple detections and controlled response relative to mission phase*
 - *Ability to account for hardware unavailability*
 - *Flexibility to address program unique requirements*
 - *FM reuse dependent on design and similarity to prior missions*
- Common weaknesses
 - *Design process did not include **model-driven engineering***
 - *Utilization of FMECA but **no formal functional failure analysis** specification or guarantee the design followed the FMECA*
 - Explicating **subtle interactions** remained a challenge
 - *No **architectural representation** suitable for decision trades from changing priorities*
 - Graceful degradation evaluation, software resource observability, constraint management, design pattern representation
 - Scalability of FM changes and their relationships to higher mission objectives
 - *FM design potential was not always realized by the implementation*
 - *Documentation often created post-design*

Better model-driven engineering practices are needed



Significant Findings: Test, Verification & Operations

- **Previous human experience critical** to test planning and test resource allocation
- **Value of SIL/HIL test bed** development/availability well understood
 - *Intended features of the FM architecture - testable*
 - *Late discovery of design/logic flaws - problematic*
- **Ability to tailor fault management on orbit** via uploadable thresholds, persistences, and hardware availability data
- **Insight into side-effects** were not always known a priori
- **Contingency planning not integral to FM architecture**
 - *Little evidence that architecture teams involved in contingency plans or ops personnel in FM design reviews*

Contingency planning could be better integrated into development practices

This chart has not been approved for public release and is subject to the restrictions on the title chart of this material.



Observations

- The FM challenges of deep space exploration and mission-centric/human-critical projects will drive architectural changes that force a rethinking of long-held design assumptions:
 - Sufficiency of fixed, predetermined, simplified view of anomaly management (suited for fail-safety) over an *adaptive*, contextually appropriate situational response (suited for fail-operational)
 - Sufficiency of ground control over *on-board autonomy* (in low BW, hi latency)
 - Recognition of context-sensitive priority deadlines vs. fixed hierarchy priority
 - Land navigation FM strategies (*goal-oriented* over component-directed monitoring)
 - Control hierarchies balancing vehicle safety and mission completion can blur *bus vs. mission separation of concerns*
 - Single fault tolerant designs vs. *multiple fault* coordinated designs
- While current FM designs continue to evolve and mature, system engineering practices have not adequately allayed concerns of how to make and manage these FM design tradeoffs

Development risk will need to be mitigated as these design alternatives mature



Recommendations

Recommendations	Benefits
Develop formal functional failure analysis specifications which are analyzable for completeness and consistency	<ul style="list-style-type: none">• Enables automated path analysis to identify potential race conditions from incorrect persistence, rate group assignments, timeliness, etc• Enables analysis of specification alignment with implementation• Evaluate fault verification sensitivity to transient conditions• Understand design robustness
Formalize analyzable architectural representations through model driven engineering practices	<ul style="list-style-type: none">• Design model representations (e.g. state generation, rule-macro dependencies) can be used to auto-generate FM detection/response logic code, rules, macros, variables• Enables design-code alignment assessment (Are you building what was designed?)• Enables dynamic discrete event simulations from architectural models to evaluate alternative design strategies (e.g. sufficiency of rate group/schedule changes)• Can define design patterns, domain-specific profiles to characterize mission-specific needs, goals, product-line architectures• Can minimize implementation errors via naming, design pattern conventions• Can evaluate design principles via model checking (e.g. rule dependency constraints)• Can evolve architectural changes through model transformations• Can leverage commercial tool integration

Green Denotes Standard Practice



Recommendations

Recommendations	Benefits
Develop system engineering (e.g. SysML) models for constraint management, use case analysis, SW FMEA management	<ul style="list-style-type: none">•Identifies off-nominal conditions, their mitigation choices to improve contingency plans•Can facilitate stress testing design•Enables analysis of software design decisions•Can model cross-cutting constraint checks and use “satisfies” traces to verify rules/code compliance (e.g. power, weight constraints)•Can identify key HW/software resources to improve software observability (e.g. watermarking)•Formalizes rationale for strategy choices
Develop reward-based models to evaluate performance/ reliability/ dependability tradeoffs for selected scenarios	<ul style="list-style-type: none">•Enables objective analysis of design choice impacts on mission accomplishment in terms of performability•Assists in assessing strategy/goal options for adaptive fault tolerant systems with graceful degradation

Yellow Denotes Preferred Practice



Recommendations

Recommendations	Benefits
Develop techniques to dynamically improve software observability of FM resources (e.g. pre-planned observer agents, potential rule fire tracking)	<ul style="list-style-type: none">•Improve implementation of context-dependent telemetry collection
Evaluate effectiveness of context dependent strategies	<ul style="list-style-type: none">•Enable autonomous context-dependent telemetry collection (e.g. dynamic rule activation/creation or preplanned layered strategies for selected SW/ HW observability)•Improve remote situation assessment
Research/evaluate design alternatives for deep-space robotic missions and apply these to architectures	<ul style="list-style-type: none">•Identify implementation techniques to moderate between deterministic and adaptive control•Evolve architectural FM design toward phase-sensitive goal-oriented strategy•Understand whether agent-based fault containment can be used in remote diagnosis•To assess effectiveness of ground-enabled/ autonomy strategy options, autonomous HW reconfiguration
Conduct multiple failure stress testing of adaptive designs	<ul style="list-style-type: none">•Determine design robustness/effectiveness of self-supervised learning strategies•Can identify stability limits of strategy options via alpha-beta gaming•Assess adaptive problem solving•Environment tested prototypes

Pink Denotes Possible Improvement



Summary

- Two fault management **architecture families were qualitatively assessed**
- Given unknown emergent behaviors, future robotic space missions (especially deep space missions) will require greater **autonomy choices that will require rethinking** of some fault management design assumptions/priorities
- Both architectures have strengths with respect to scalability of emergent behavior, but emphasize **different FM paradigms**
- While an architecture may employ an advantageous software approach, the challenges to fault management remain a systems engineering challenge



Backup

This chart has not been approved for public release and is subject to the restrictions on the title chart of this material.



Summary Details

- Two fault management **architecture families were qualitatively assessed**
 - *Both architectures have mature heritage and provide varying degrees of fail-safe fault protection but both architectures will need to evolve to provide reliable goal-oriented, context-sensitive capabilities demanded by future robotic space mission explorations*
 - *Site interviews elucidated the need to maintain mission through critical orbit insertion maneuvers as well as mission-specific surface operations. Several software development practices were also emphasized.*
- Given unknown emergent behaviors, future robotic space missions (especially deep space missions) will require greater **autonomy choices that will require rethinking** of some fault management design assumptions/priorities
- Both architectures have strengths with respect to scalability of emergent behavior, but emphasize **different FM paradigms**
 - *A: Exploits a virtualized component paradigm that can closely map to HW configurations but will need to expand and integrate system level control strategies to understand goal-directed behaviors in the presence of fault conditions. Although the architecture has the potential to do so, no current implementation supports this integration. FM without of situational context will have limited effectiveness. Performance of layered self-organizing strategies will need to be studied.*
 - *B: Uses a rule based control paradigm which can be flexibly organized into prioritized/hierarchical rule sets. Although rule sets can be organized into goal-oriented strategies, their effective use faces many challenges with rule contention, conflict resolution, distraction. Performance of interpreted rule schemes will need to be re-evaluated as more context sensors are needed to be assessed at rates higher than 1Hz.*
- While an architecture may employ an advantageous software approach, the challenges to fault management remain a systems engineering challenge



NASA Fault Management Workshop Recommendations

	Recommendations
1A	Allocate FM resources and staffing early , with appropriate schedule, resource scoping, allocation, and prioritizing. Schedule V&V time to capitalize on learning opportunity.
1B	Establish Hardware / software / “sequences” /operations function allocations within an architecture early to minimize downstream testing complexity.
1C	Engrain FM into the system architecture. FM should be “dyed into design” rather than “painted on.”
2A	Establish clear roles and responsibilities for FM engineering.
2B	Establish a process to train personnel to be FM engineers and establish or foster dedicated education programs in FM
3	Standardize FM terminology to avoid confusion and to provide a common vocabulary that can be used to design, implement, and review FM systems.
4A	Identify representation techniques to improve the design, implementation and review of FM systems
4B	Establish a set of design guidelines to aid in FM design
5A	Identify FM as a standard element of the system development process.
5B	Establish metrics that will allow proposal evaluators and project teams to assess the relevance, merits, and progress of a particular FM approach
6A	Design for testability: Architectures should enable post-launch and posttest diagnosis.
6B	Examine all observed unexpected behavior.
6C	Implement continuous process improvement for FM lifecycle.
6D	Catalog and integrate existing FM analysis and development tools, to identify capability gaps in the current generation of tools and to facilitate technology development to address these gaps.
7	Review and understand the impacts of mission-level requirements on FM complexity. FM designers should not suffer in silence, but should assess and elevate impacts to the appropriate levels of management.
8	Assess the appropriateness of the FM architecture with respect to the scale and complexity of the mission and the scope of the autonomy functions to be implemented within the architecture.
9	Define and establish risk tolerance as a mission-level requirement.
10	Examine claims of FM inheritance during the proposal evaluation phase to assess the impacts of mission differences.
11	Develop high-fidelity simulations and hardware testbeds to comprehensively exercise the FM system prior to spacecraft-level testing
12	Collect and coordinate FM assumptions, drivers, and implementation decisions into a single location that is available across NASA, APL, and industry. Utilize this information to establish / foster dedicated education programs in FM



Assessment Criteria and Recommendations Map

Area	Fault Management Workshop Recommendations																			
	1A	1B	1C	2A	2B	3	4A	4B	5A	5B	6A	6B	6C	6D	7	8	9	10	11	12
Planning																				
Clarifies organization of resources and roles	X			X			X		X				X				X			X
Balances flexibility and complexity in early design choices and test effort impacts										X					X	X				
Requirements																				
Addresses program unique mission requirements to provide a reliable, safe FM capability		X	X												X					
Considers the entirety of FM functionality when allocating requirements to HW, SW and operations		X	X																	
Design																				
Is well coordinated and appropriate to the mission needs	X				X	X										X		X		
Utilizes architectural representation and analysis techniques that improve the design, implementation and mitigate risk			X				X	X			X		X				X			
Manages system complexity of robotic deep space missions and considers spacecraft robustness early in the design			X					X							X					
Implementation																				
Facilitates reuse								X								X		X		
Easily adapts to change	X		X												X					
Demonstrates reliable FM capabilities/features for specified mission								X							X	X	X			
Test and Verification																				
Facilitates test design and execution	X										X									X
Identifies the level of risk exposure			X									X				X				X
Operations																				
Provides understanding of operational impacts on design and implementation				X																
Facilitates ease of operations use							X			X					X					



Sample Evaluation Questions

Planning	Related Questions/Concerns
Clarifies organization of resources and roles	<p>Did the FM design process begin early (e.g. PrePhase A)?</p> <p>Were FM engineers involved early in planning? (Was FM engineering a separate group?)</p> <p>Were experienced team leads in place throughout the project?</p> <p>Were there efficient allocation of resources (e.g. Did the program plan execute as planned or were effort estimations too optimistic?)</p> <p>Was there an understanding of what the risk tolerance of the mission would be?</p> <p>Did the FM architecture develop with clear relationships with other project tasks (e.g. systems engineering, safety, mission assurance)</p> <p>Did the process permit early discovery/resolution of issues?</p> <p>Any continuous process improvement practices?</p> <p>Any mismatch in expectations, priorities, scope within the organization?</p> <p>Any lack of disciplined practices?</p>
Balances flexibility and complexity in early design choices and test effort impacts	<p>Any notable issues between flexibility and complexity? E.g., were critical behaviors unnecessarily complex or did the architecture employ simplicity where possible?</p> <p>Were quality attributes/metrics used to assess complexity and completeness of architecture? (e.g. diagnostic coverage of fault space, time responsiveness of faults, TPMs, fault dependency analysis to identify design gaps/risk?)</p>
Requirements	
Addresses program unique mission requirements to provide a reliable, safe FM capability	<p>Were the properties of the elements (HW, SW, ops) employed to protect the system understood so the design could fulfill program-unique FM, reliability and redundancy requirements</p> <p>Were FM requirements identified separately?</p> <p>Was FM considered throughout the project or toward the end?</p> <p>Did conflicting requirements or poor design decisions lead to higher FM complexity?</p>
Considers the entirety of FM functionality when allocating requirements to HW, SW and operations	<p>Were FM requirements allocations to HW/SW/operations made early to facilitate testing?</p> <p>Does the FM architecture logically flow from the requirement allocations?</p> <p>Were adequate use cases defined that covered critical operational scenarios and behavioral sequences?</p>



Sample Evaluation Questions

Design

Is well coordinated and appropriate to the mission needs

- Did FM terminology foster miscommunication?
- Were heritage reuse assumptions appropriate to the design?
- Were new FM approaches considered?
- Were FM design assumptions, mission drivers, decisions well coordinated?
- Any V&V impacts from heritage reuse?
- Were lessons learned shared across related programs?

Utilizes architectural representation and analysis techniques that improve the design, implementation and mitigate risk

- Was model-driven engineering practiced? (e.g. architecture-centric vs code-centric development)
- How was the architectural design represented? (UML, state diagrams, flow charts?) Was it complete, accessible, and meaningful to all S/C disciplines?
- How usable are the design artifacts for assessing future mission capabilities? Maintenance?
- Were fault tree and FMECA analyses performed and did this improve the architecture?
- Any studies, simulations, prototypes made to assess system performance?
- How well does the architecture minimize conflicts in detection, isolation, response? Any analysis performed to eliminate or minimize conflicts in detection, isolation, and response? (Any tool usages for gap analysis?)
- Were fault responses decoupled to reduce testing complexity?
- Were fault responses evaluated for timeliness/situational context to avoid premature resource swapping
- Was SW FMEA performed? Could the architecture easily support SW FMEA mitigations?
- Was architecture representation sufficient to make decision trades from changing priorities (e.g. cost-based trades vs risk-based trades)?

Manages system complexity of robotic deep space missions and considers spacecraft robustness early in the design

- Any evidence of spacecraft robustness early in design?
- Does the architecture provide both autonomous and ground action support?
- Support for both fail-safe and fail operational requirements?
- Was a safety net provided in which the FM will fail-safe every situation even if the fault is not specifically identified a priori?
- Does FM design ensure complete fail-safety via dead-ending?
- Is the design flexible to changes in HW, SW, or operations? (Was this appropriately balanced against complexity of robotic missions?)
- Does the FM design facilitate implementation and test?

Implementation

Facilitates reuse

- Did the FM implementation enable elements to be reused? (e.g. software components, facilitated HW reuse, facilitated design reuse)

Easily adapts to change

- How well does the FM design permit the implementation to easily adapt to change?
- Were there any unforeseen development changes? If so, how well did the implementation support those changes?
- Were any design oversights or mission changes able to be accommodated after launch?
- Did the implementation reflect a good balance between complexity and flexibility?

Demonstrates reliable FM capabilities/features for specified mission

- How did the FM implementation fair on-orbit? (E.g., Any notable strengths or shortfalls?)
- How complete was the fault detection/response isolation? Any unexpected cascading of fault conditions?



Sample Evaluation Questions

Test and Verification	
Facilitates test design, execution	Were testbed resources adequate?
	Were high-fidelity simulations, component behavioral models (e.g. SW in the loop) and HW in the loop testbed environments developed?
	Did FM architecture support fault injection during simulation studies and verification?
	Were initial conditions varied to understand their effects? (e.g. across phase or mode changes?)
	Did the FM architecture simplify the testing of requirements?
Identifies level of risk exposure	Was a scenario-based "Test Like You Fly" context supported?
	Were test cases complete?
	Any stress testing/off-nominal testing performed
	Was some measure of test coverage determined so the level of risk exposure could be identified?
	Were all observed unexpected behaviors examined and anomalies adjudicated?
Operations	
Provides understanding of operational impacts on design and implementation	Did FM development process facilitate communication of operational impacts?
	Were contingency plans complete and documented?
Facilitates ease of operations use	Was support provided for FM state enable/disable (e.g. detection, response) and ability to adapt to failed units or use redundant hardware
	Did FMS design/implementation provide the proper balance between autonomous vs ground supported anomaly resolution? (e.g. the ability to adapt to failed units or use redundant HW)
	Any early indicators of problems provided vs. waiting till verified fault?
	Were operators provided the ability to modify/calibrate FM while on orbit?
	Was adequate visibility into anomalous behavior provided via telemetry and stored data? (e.g. rapid diagnosis following a fault)
	Any support for monitoring SW resources (e.g. queue sizes, buffers, state settings, etc)

