Software Design Policies for a Large-Scale Command and Control Software Development Project

Lee Beeman, Michael Bergan, William Dugan, Race Filarey, Nancy Myers, John F. Reeves, Jim Zalinski

TRW Systems Information and Technology Group
One Space Park, Redondo Beach, California 90277
Software Design Policies

- Define the rules and procedures for realizing an architecture in a completed system
- Maintain consistency in design across multiple, parallel development efforts
- Provide mechanisms for architects to
  - Maintain correspondence between architecture and development
  - Evaluate the architecture as implemented
  - Influence the development to address system quality features
Example: Mission Specific Class Policy

- **Motivation:** Design object class hierarchies for reuse on future C&C programs
- **Approach:** Decompose design to isolate mission specific dependencies
  - Capture mission specifics in data, not code
  - Use polymorphism to normalize mission dependent interfaces
- **Policy** describes common mission specific cases, and how to design for reuse
Mission Specific Class Policy

Characteristics

• Strategically motivated
  – Strives to position code for reuse without a specific reuse target

• Focus on new design technology
  – Describes application of OO techniques mission specific class design

• Describes criteria for definition of mission specific classes
  – Checklist of design criteria that must be present
Project Background

• Policies developed for the Data Acquisition and Control System (DACS)
  – Mission critical command and control system
• Four year development, 100+ software engineers
  – Incremental lifecycle, OOA&D, C++, High reliance on COTS
• Driving real-time performance and availability requirements
  – ATM Network of Unix Workstation, CORBA ORB middleware
DACS Design Policies

- Data Storage
- System Control
- Interfaces and Messaging
- Anomalous Condition Handling
- Data Collection for Test Verification
- Portability

- Naming Conventions
- Constants, Units, and Types
- Unix Process Architecture
- Object-Oriented Modeling
- Architectural Layering
- Multi-Threading
- Mission Specific Classes

GSAW99
Motivation for Design Policies

- Infrastructure and COTS
- Project staff background and experience
- Customer focus
- Strategic direction

Policies address areas where there is risk of divergence during development
Policies for Physical Architecture Risk Areas

- Data Storage - Addresses performance risks associated with new technology (OODBMS)
- System Control - Addresses reliability, availability and performance risks of Unix process architecture
- Interfaces and Messaging - Addresses performance and maintenance risk of IDL interfaces
- Portability - Addresses risk of platform dependence
- Unix Process Architecture - Addresses performance risk of C++ and Unix process definition
Format of Design Policies

- **Rationale** - Identify quality criteria, beneficiaries, risks addressed
- **Policy Statement** - The system design, constraints on development, and how the design satisfies the policy rationale
- **Software Requirements** - How the design impacts software development

Policies must include process description: what, where, when and how
Characteristics of Successful Software Design Policies

• Motivated - Describes how the system is better, and who it is better for
• Measurable - Provides visible evidence of policy in the implemented system
• Evaluatable - Describe expected consequences of policy adoption
• Automatable - Reduce policy overhead on developers and architects
• Flexible - Describe how the system and policy can change over time