Architectural-Centric Representation for Design Diversity and Program Evolution

Phillip Schmidt, Ph.D.
The Aerospace Corporation
Phillip.P.Schmidt@aero.org

Sergio Alvarado, Ph.D.
The Aerospace Corporation
Sergio.Alvarado@aero.org

Jesus Rivera
The Aerospace Corporation
Jesus.L.Rivera@aero.org

Jaime Milstein, Ph.D.
The Aerospace Corporation
Jaime.Milstein@aero.org

Agenda

- Definitions
- Experience with Architectural Representations
- Current Approaches to Manage Evolution & Design Diversity
- Software Architecture Representation Analysis and Experimentation Environment (SARAEE)
- Closing Comments

Breakout Topics Addressed

- 1. Architecture as blueprint
- 4. Architecture representation
- 9. Architecture as a tool to manage change

Definitions

- Design diversity, or more generally, architectural variability, refers to the ability to identify and flexibly reshape aspects of an architecture
  - Aspects identify points of variation
- Program evolution refers to the ability of an architecture, over its lifecycle, to undergo change

Experience with Architectural Representations

- Vision: Architecture is central to supporting design diversity and program evolution
  - Many methodologies and techniques available
- Reality: Software architectural representations often incomplete and inconsistent
  - Tools, training issues
    - New (untried) technologies
    - Multiple skill levels
  - Commitment issues
    - New (uncomfortable) processes
  - Complexity issues
    - Disconnected artifacts
    - Multiple formats
- Disconnect between vision and reality

Current Approaches for Managing Evolution

- Manage change at source code level (status quo)
  - Not efficient
  - Impact analysis using code + available SW engineers is only 40-50% effective in identifying necessary changes [Lindvall]
    - Greater dependency on human expertise, peer reviews
    - Lower effectiveness with cross-cutting changes and interfering changes
    - Diminishing returns as complexity increases
  - Cost of managing change at code level is higher (at least an order of magnitude) than during architectural design [Lientz],[Bengtsson]
  - Exploring alternative architectural solutions is narrower, less insightful
    - Eventually leads to architectural drift and poorly understood systems [van Gurp]
    - Precludes early (pre-code) analysis
  - Impacts due to new languages can be costly

Current Approaches for Managing Evolution (cont'd)

- Improve architectural representation
  - Reduces risk of failure to meet quality requirements prior to building it
  - Architectures without behavioral information only 40% changes visible
  - Architectures with behavioral (e.g. method bodies) information 80% changes visible [Lindvall]
- Anticipate change better
  - Product line emphasis
  - Anticipating potential change costs less during architectural design
  - Abstracted conceptual context with versioning is needed to understand asset dependencies in new contexts [Bosch]

Current Approaches for Managing Design Diversity

- Scenario-based evaluation
  - Checklists
  - Scenario-based tradeoff methods [Kazman]
- Feature-oriented variability
  - Feature graphs (variability as delayed binding)
  - Separation of concerns
    - Use cases,inheritance, composition, configuration scripts, reflection, naming directories, conditional compilation
- Architectural-centric representation
  - Reuse-contract management [Mens]
  - Architecture definition languages supporting Component Variability, Product line versioning [xadl]
- Aspect-oriented programming
  - Specify aspects within the programming language space
- Aspect-oriented architectural assessment
  - Specify aspects within the architectural space

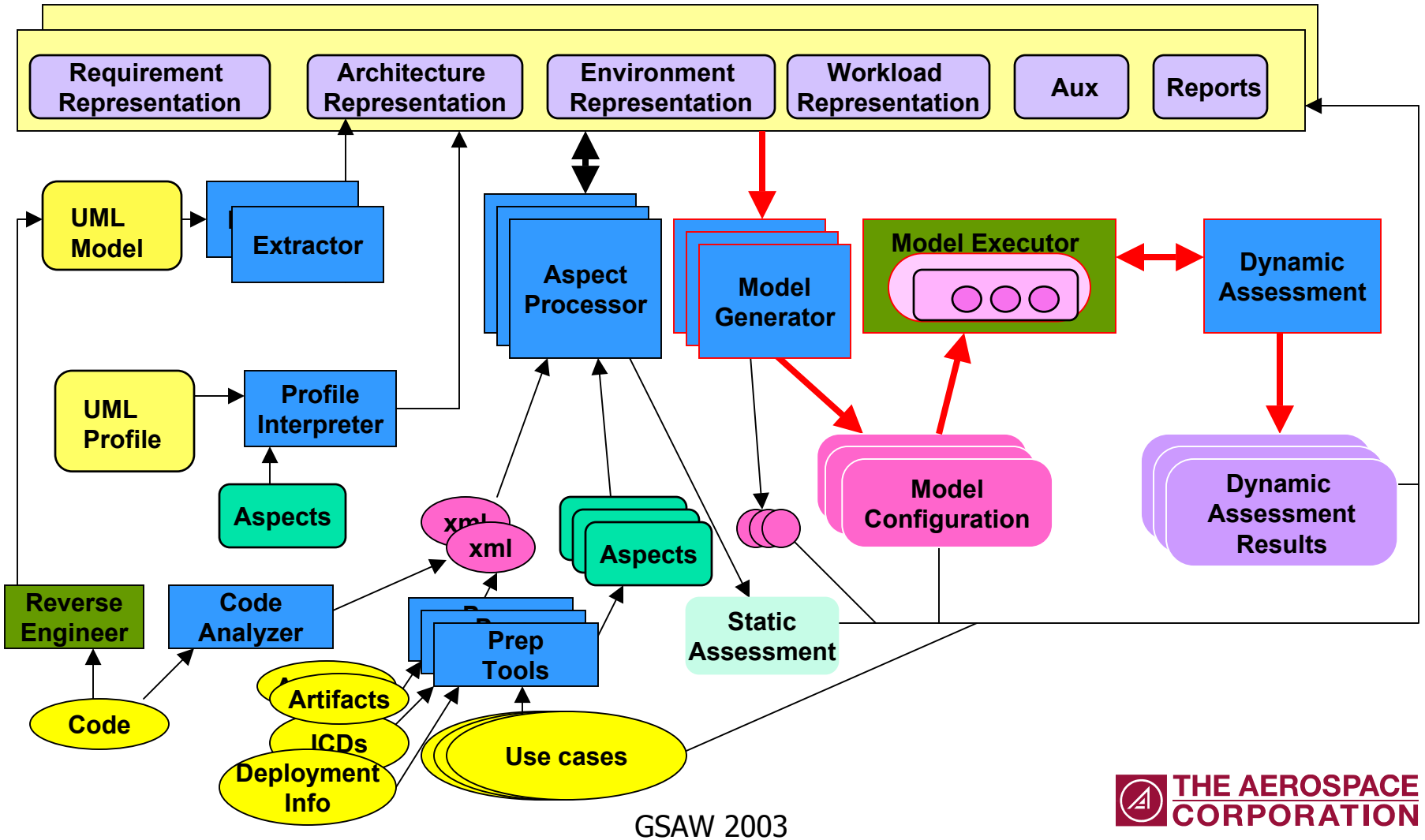
Observations

- Ill-defined architectural models likely to persist
  - Incompleteness and inconsistencies will continue
  - Not all features will be preplanned and separable
  - Architectural complexities/dependencies will make feature interactions difficult to manage
- Architectural evolution is a shared responsibility
  - Need to analyze the architecture not only to identify shortfalls of the system being built, but also to cope with the practices that caused and perpetuate the shortfalls in the first place
  - Need better ways to convey architectural context, and discover, communicate and respond to architectural concerns
- Architecture is a core asset
  - Need to improve architectural representation from various artifacts
  - There are concerns that code alone cannot answer

SARAEE Objectives

- Improve representation of features and concerns
  - Any given OO decomposition will eventually be reexamined
  - Need to look backward, forward, and elsewhere! (e.g. old design decisions, new usage scenarios, other ICDs)
  - Expand features to study concerns we don't want! (e.g. design conflicts, deadlocks, unreachable states)
  - Manage complex, cross-cutting, interdependent interactions
- Recognize representations have a context and semantics
- Recognize representations must support frequent change
- Support aspect-oriented architectural analysis using multi-level modeling
- Ensure compatibility with other approaches

SARAEE Approach



SARAEE Approach (cont'd)

- Develop tools/techniques to improve context and semantics
  - XML schemas represent/share architectural artifacts
  - Support augmentation from various sources
  - Support interpretation aspects (e.g. UML profiles of use)
- Augment representations with parameters derived from reverse-engineered code
  - Capture missing behaviors to improve evolution success
- Manage planned scenarios as analyzable use cases
- Manage planned features as aspects over entire representation space
  - Dependencies too difficult otherwise
- Move toward automating analysis and aspect-oriented impact analysis
- Develop architectural analysis techniques to discover design patterns and refactoring opportunities

Closing Comments

- Improving architectural representation is cost-effective for managing change
- A product-line feature-oriented architectural perspective is helpful for managing planned evolution and supporting design diversity
- Need expanded support for defining concerns, managing feature interactions, and clarifying architectural context
- Aspect-oriented architectural analysis could close the representational gap
- SARAEE is a step in that direction

References

- **Scenario-based**
  - R. Kazman, G. Abowd, L. Bass, P. Clements, " Scenario-Based Analysis of Software Architecture," *IEEE Software*, 13 (6):47-56, 1996
  - R. Kazman, M. Klein, M Barbacci, T. Longstaff, H Lipson, J. Carriere, "The Architecture Tradeoff Analysis Method," in Proceedings of the 4th International Conference on Engineering of Complex Computer Systems (ICECCS98), Monterey, CA, IEEE CS Press, pp68-78, 1998
  - P. Bengtsson, N. Lassing, J. Bosch, H. vanVliet, "Analyzing Software Architectures for Modifiability," May 2000
- **Feature-oriented**
  - M. Svahnberg, J. Van Gurp, J. Bosch, "On the Notion of Variability in Software Product Lines" *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA 2001),* pp 45-55, August 2001
- **Architecture-centric Design**
  - T. Mens, C. Lucas, P. Steyaert, "Supporting Disciplined Reuse and Evolution of UML Models," *First International Workshop on UML*, Mulhouse, France, June 1998
  - Xadl, an XML architecture description language http://www.isr.uci.edu/projects/xarchuci/

References (cont'd)

- **Aspect-oriented Programming**
  - See *Communications of ACM*, October 2001, Vol 44, No 10.
- **Aspect-oriented Architectural Analysis**
  - P. Schmidt, R. Duvall, G. Mulert, J. Milstein, J. Rivera, "Aspect-Oriented Architectural Analysis using Multi-level Modeling of Complex Systems," *Proceeding of 2003 International Information Resources Management Conference*, May 2003
- **Maintenance/Product Line Studies**
  - J.Bosch "Product-Line Architectures in Industry," *Proceeding of the 21st International Conference on Software Engineering*, Nov 1998
  - J. van Grup, J. Bosch, "Design Erosion: Problems and Causes," *Journal of Systems and Software*, November 2001.
  - M. Lindvall, K. Sandahl, "How well do Experienced Software Developers Predict Software Change?", *Journal of Systems and Software,* vol 43, no 1, pp 19-27, 1998
  - M. Lindvall, M. Runesson, "The Visibility of Maintenance in Object Models: An Empirical Study," *Proceedings of International Conference on Software Maintenance*, Los Alamitos, IEEE CS Press, pp 54-62, 1998
  - B. Lientz, E Swanson, *Software Maintenance Management*, Reading, MA, Addison-Wesley, 1980.