An Aspect-Oriented Strategy for Evaluating Software Architectures that Evolve

2 March 2005

Phillip Schmidt
Phillip.P.Schmidt@aero.org

THE AEROSPACE CORPORATION

Agenda

- Motivation
- Some current strategies
- New aspect-oriented (AO) strategy
- Comparison with scenario-based approaches
- Experiences applying the strategy

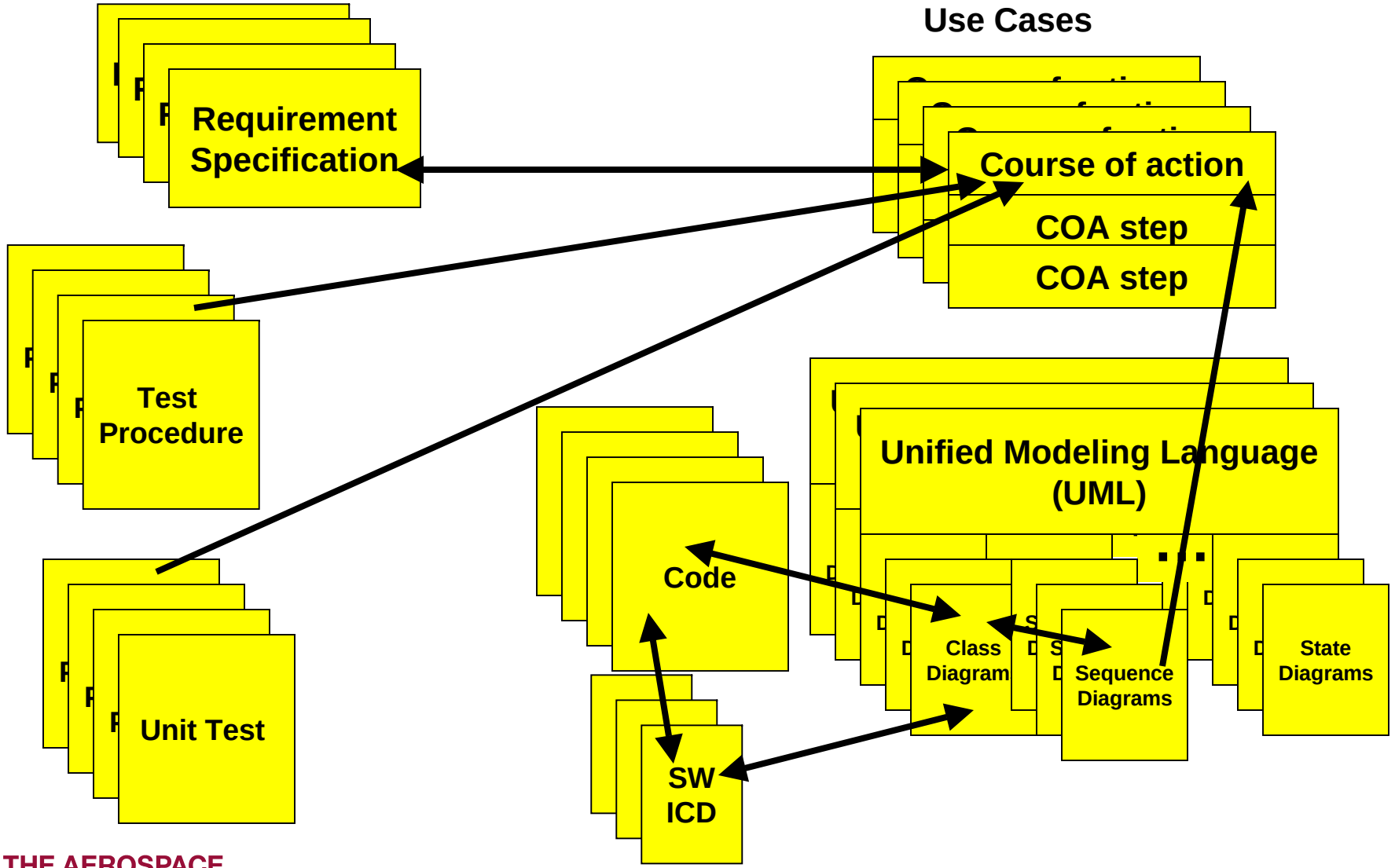
Architectural Value

- Your architectural value will be reflected in how you perceive architecture:
  - Strategic tool?
  - Something in the minds of competent people?
  - General preliminary design only, to be discarded after coding
  - Merely drawings or documentation?
  - An accurate, evolving reflection of the design?
  - Something analyzable
  - Something executable
- Perceptions have consequences and risks
- Choose wisely

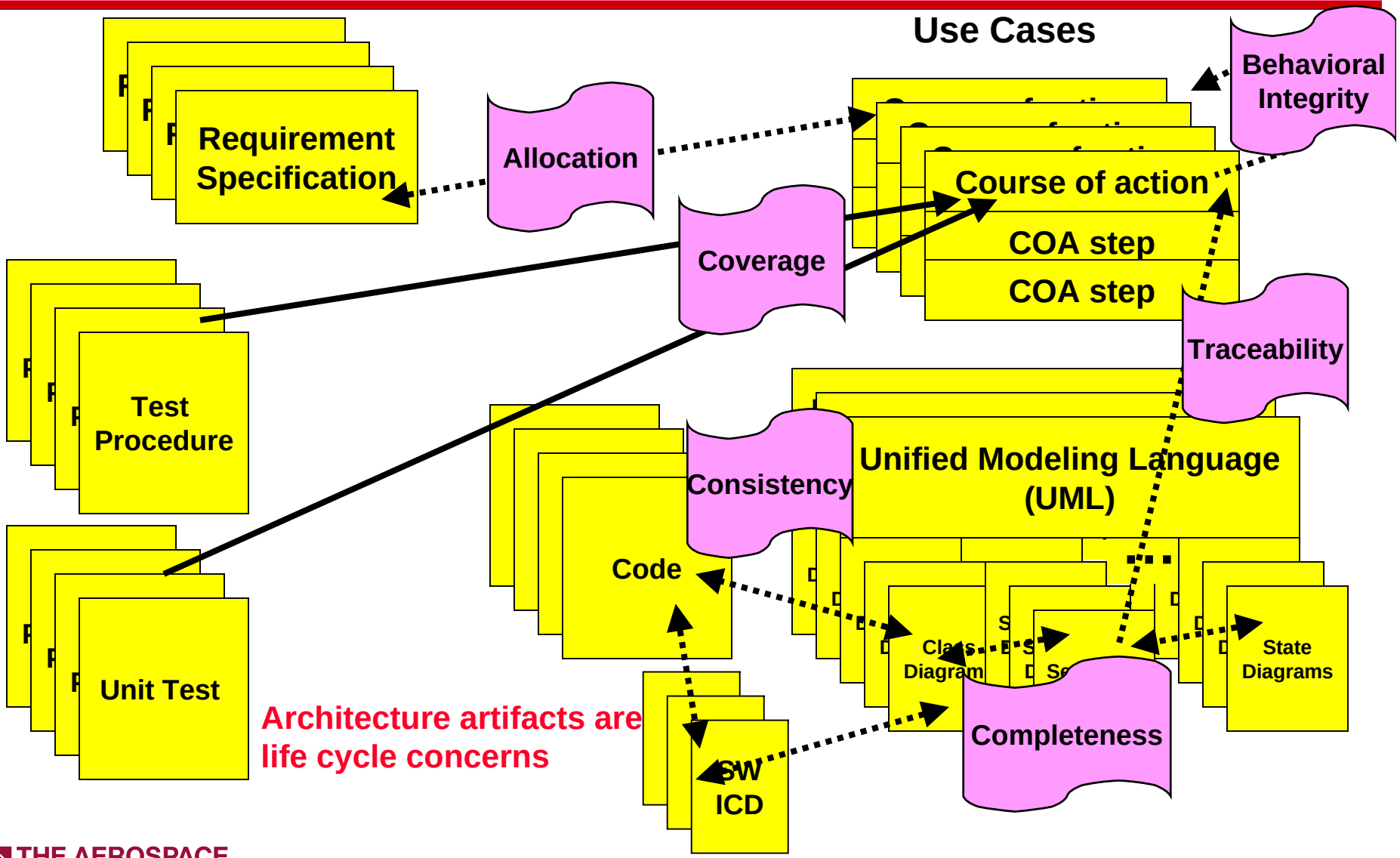
Architectural Needs

- Do your architectural values address your problems?
- Do you know you *have* problems?
- Is there an architectural value gap between you and your architectural provider?
- Architectural values are stressed when:
  - Managing complex, evolving architectural relationships
  - Architectural modeling practices are unclear
  - Unexpected concerns arise

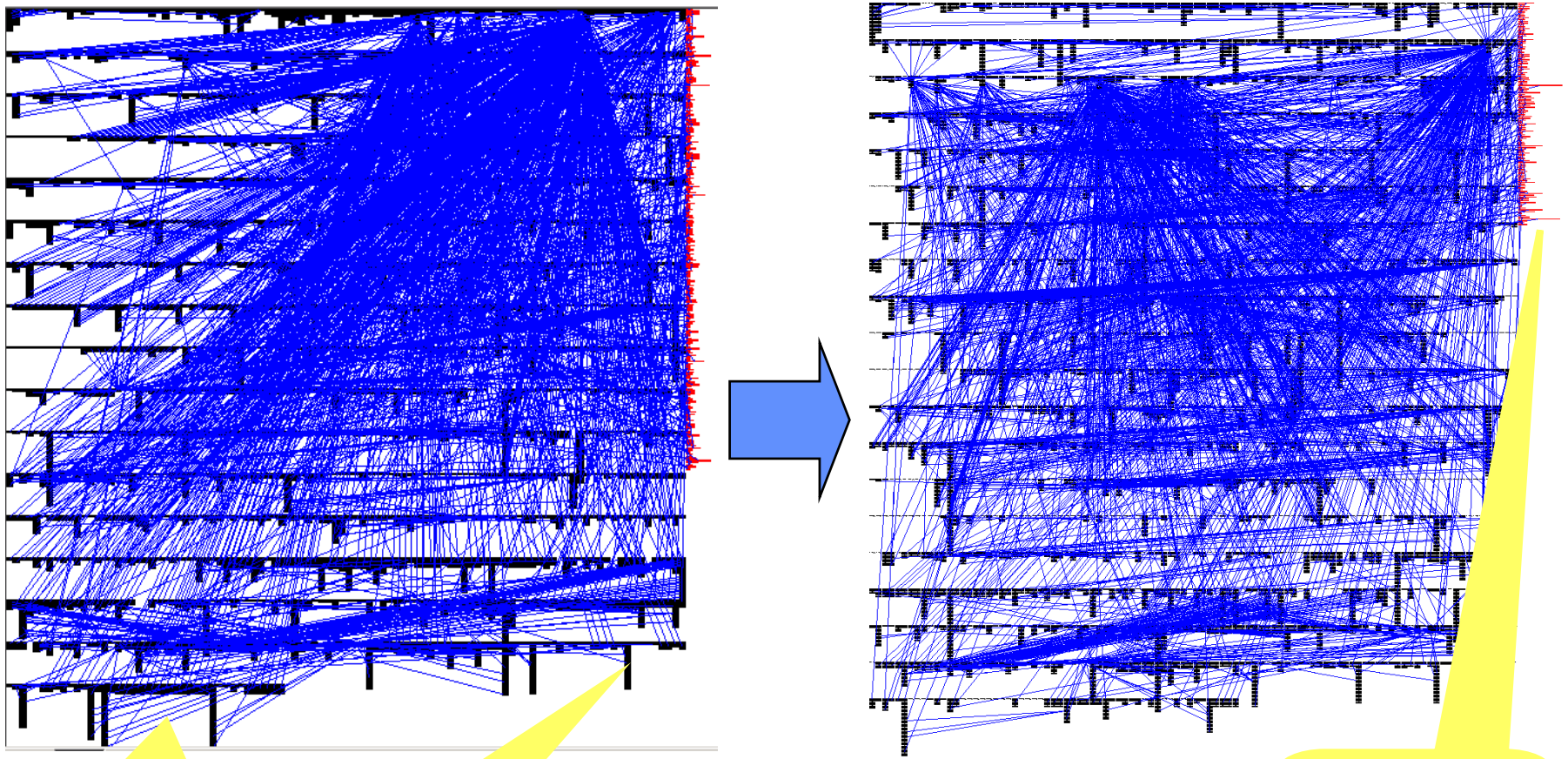
Architectural Artifact Relationships



Architectural Artifacts



Complexity of Requirement Evolution



THE AEROSPACE CORPORATION

Architectural Modeling Practices

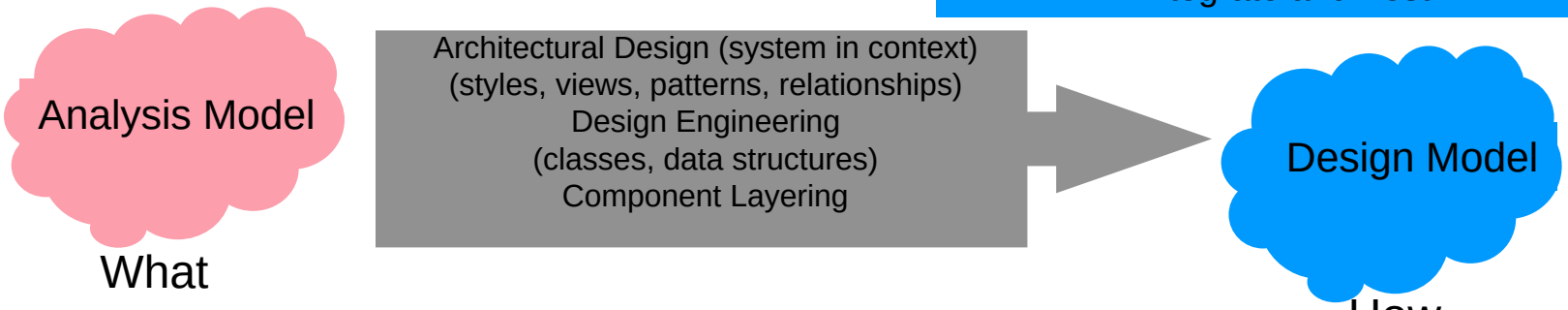
(What we think we want)

System Engineering Activities

Define assumptions
Identify simplifications, tradeoffs
Identify limitations
Define system constraints
Identify preferences
Identify, analyze and manage requirements, specifications

Software Engineering Activities

Evaluate design tradeoffs
Define concrete classes, interactions, relationships, states, activities
Identify infrastructure
Develop prototypes, deployment views
Identify product constraints, timing
Ensure traceability to requirements
Integrate and Test



What

System Modeling (abstracted world view use cases)
Informational models (data flow)
Functional models (activity flow)
Behavioral models (events, sequences)

Abstract

Problem domain oriented

How

Coherent, thorough, well planned representations

Traceable to requirements,

Completeness, correctness

Concrete realizations

Solution domain oriented

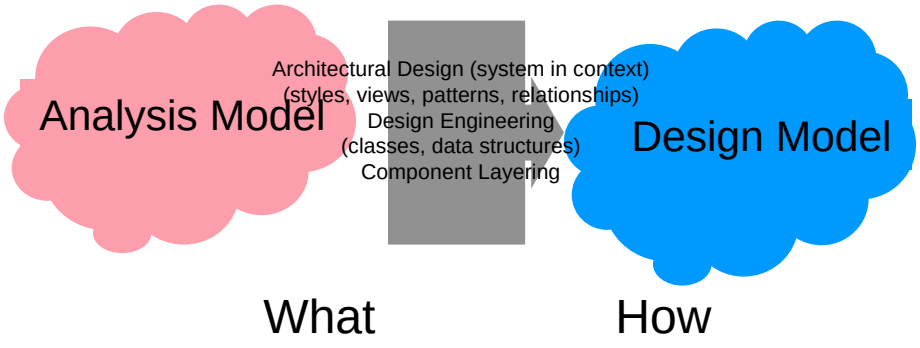
Architectural Modeling Practices
(What tends to happen)

System Engineering Activities

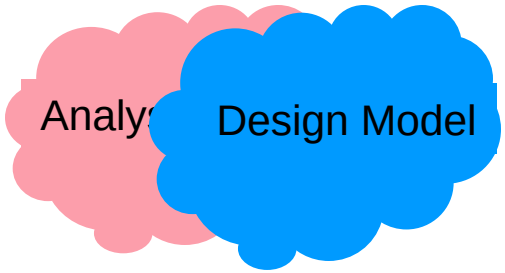
Define assumptions
Identify simplifications
Identify limitations
Define constraints
Identify preferences
Identify, analyze and manage requirements, specifications

Software Engineering Activities

Evaluate design tradeoffs
Define concrete classes, interactions, relationships, states, activities
Identify infrastructure
Develop prototypes
Identify preferences
Ensure traceability to requirements
Integrate and Test

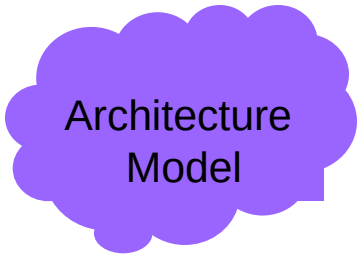


Architectural Modeling Practices
(Blurring of models)



Architectural Modeling Practices

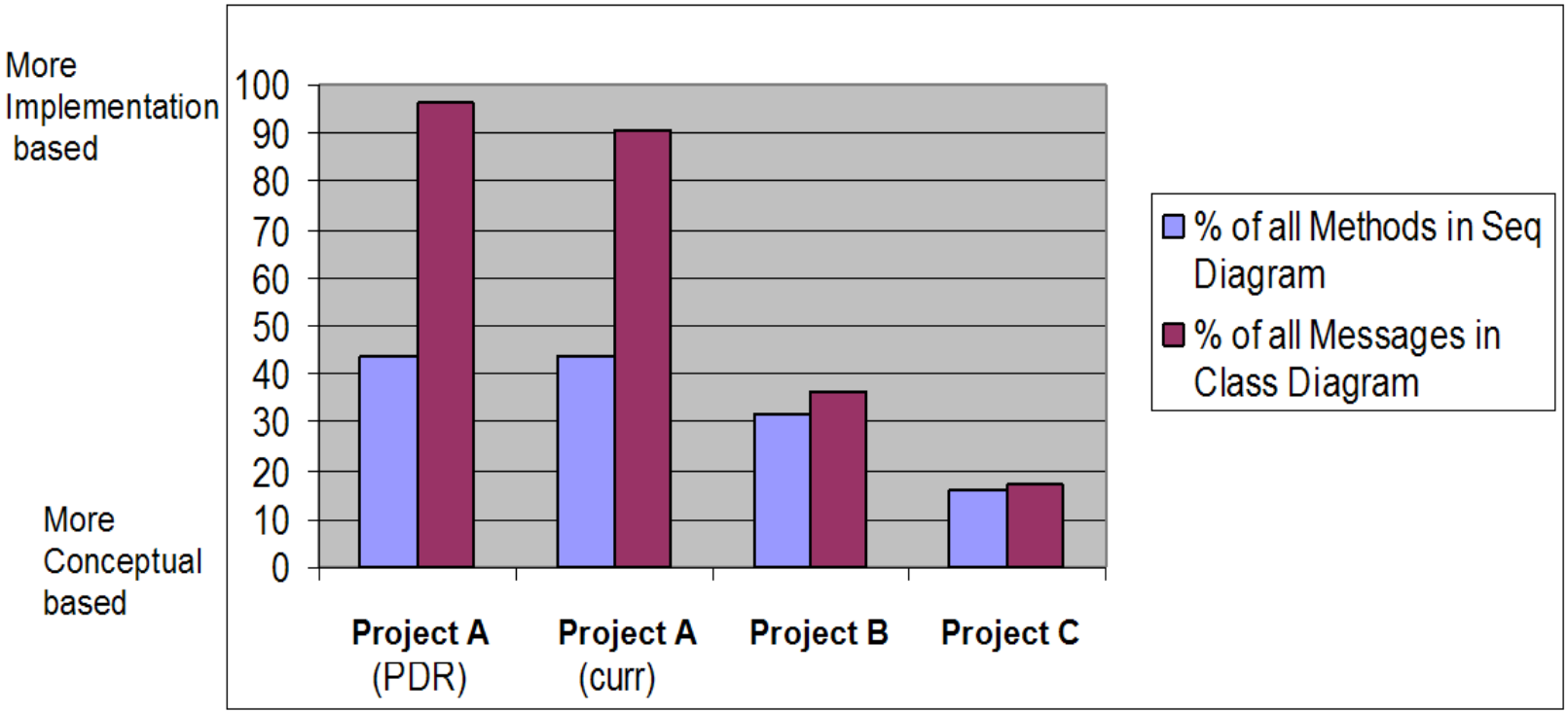
(What we usually get)



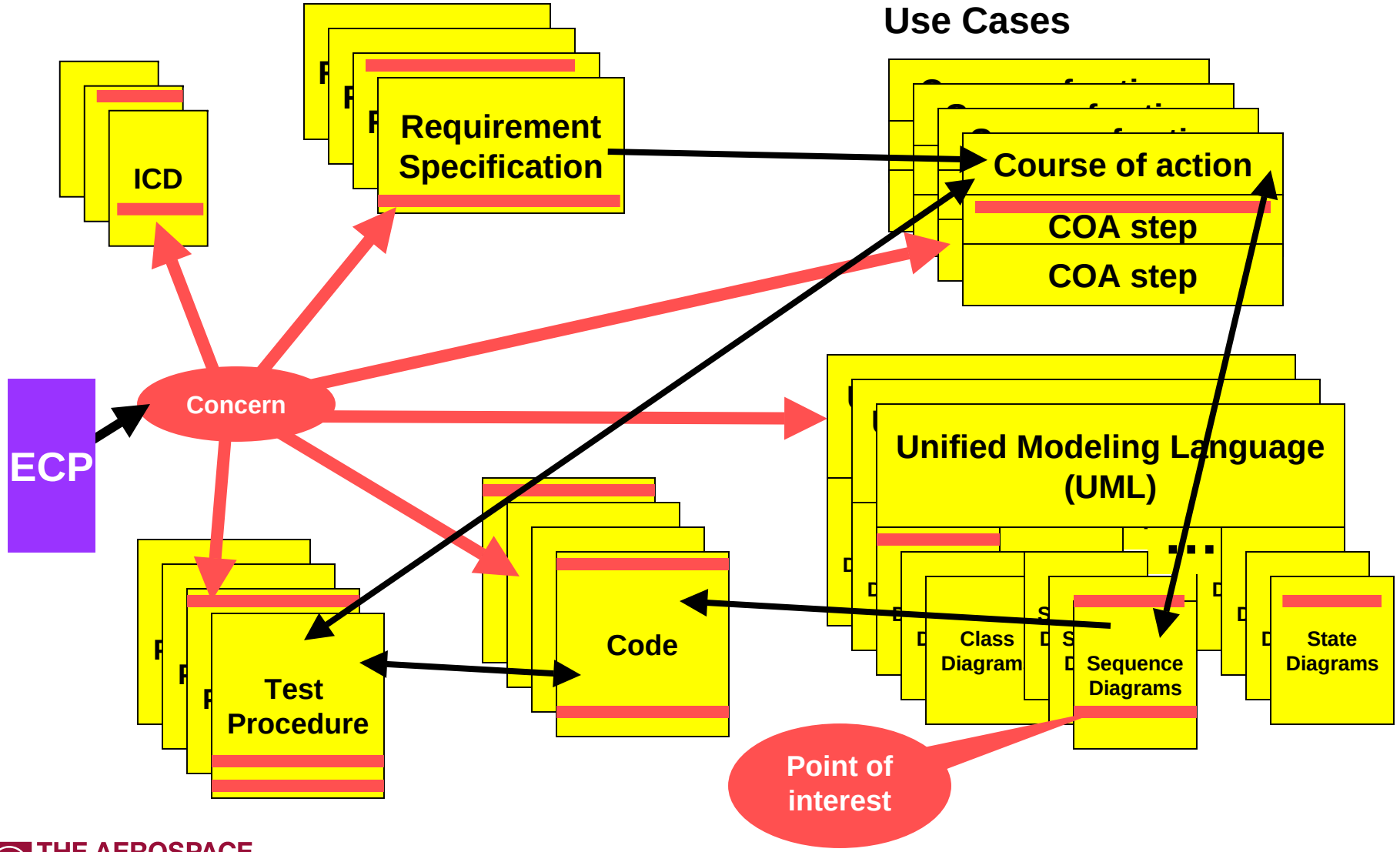
Huh?

Level of Architectural Detail

- Architectural models tend to be a mix of conceptual and implementation based information



Impact Assessment of Crosscutting Concerns



Good Intentions

- Even the “best” of architectural values can be overwhelmed by the complexity of our problems

Software Architecture Realities

- We don't know everything
  - What we know may be correct, but irrelevant
  - What we do know, may not be represented clearly
  - Scalability of manual techniques
- We can't predict everything
  - Things change unexpectedly
  - Unplanned feature interactions
- We tend to ignore the hard stuff
  - Non-functional requirements
  - Conflicting operational concepts/goals
  - Domain-specific details within commercial tools
  - Subtle software-hardware real-time dependencies

Space architectures are handicapped by their complexity

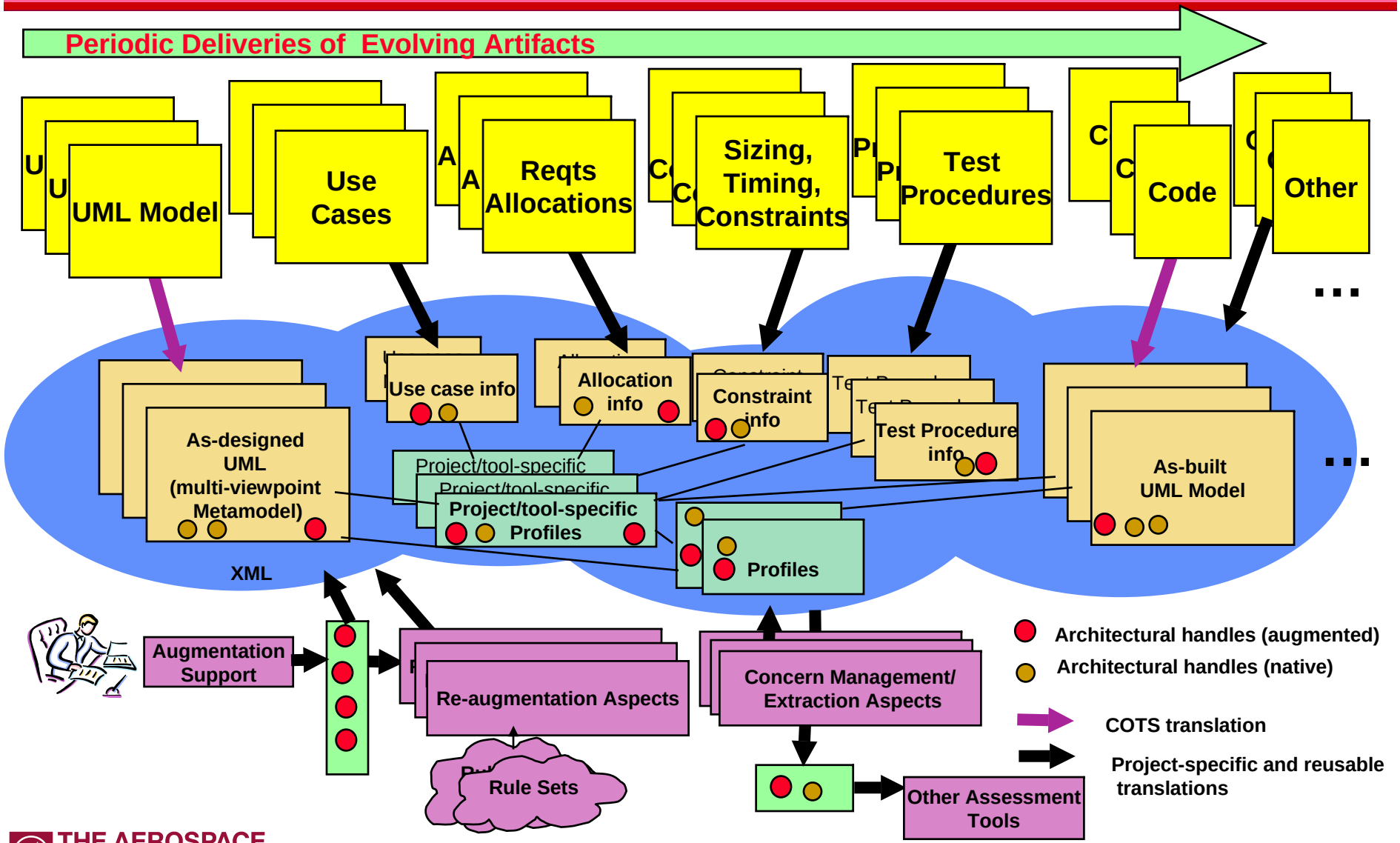
Strategies for Managing Handicapped Architectures

- Ignore It:
  - It's the contractor's problem.
- Contract It
  - Just put the required level of detail on contract
- Enforce It
  - Have contractor develop the appropriate level of architectural granularity
- Evangelize It
  - Motivate best practices through conferences, tutorials
- Analyze It
  - Stakeholder collaboration for effective automated analysis

Strategic Advice

- Develop analyzable architectures
- An aspect-oriented strategy can help with assessing evolving, handicapped architectures that might not be natively analyzable.
- Reduce architectural value gaps through aspect-oriented augmentation

Aspect-Oriented Assessment



Comparison of Scenario-based and Aspect-Oriented SW Assessments

Scenario-based Strategy	Aspect Oriented Strategy
Quality attributes, prior to development interpreted in context of pre-planned provided scenarios. Feature management in terms of planned change.	Planned or unplanned concerns described as aspects of interest over an architecture. Crosscutting concerns support unplanned changes
Predictive strategy. E.g. quality attributes depend on some pre-defined scenario context.	Corrective strategy. Some quality attributes do not have a scenario context that can be predicted. Concern evaluation assumes an architectural representation.
Questioning technique with reliance on human consultation for completeness of scenario interactions; qualitative metric formulation. Good for conceptual overview of whole system	Managing complex interactions require measurement and evaluation. Quantitative emphasis on completeness. Good for investigating issues raised from questioning techniques
Use of taxonomic hierarchies to define quality attributes	Uses architectural profiles to support augmentation, derivation, monitoring, and evaluation of architectural aspects of interest.
Phased application	Periodic application during life cycle

Applying an Aspect-Oriented Strategy

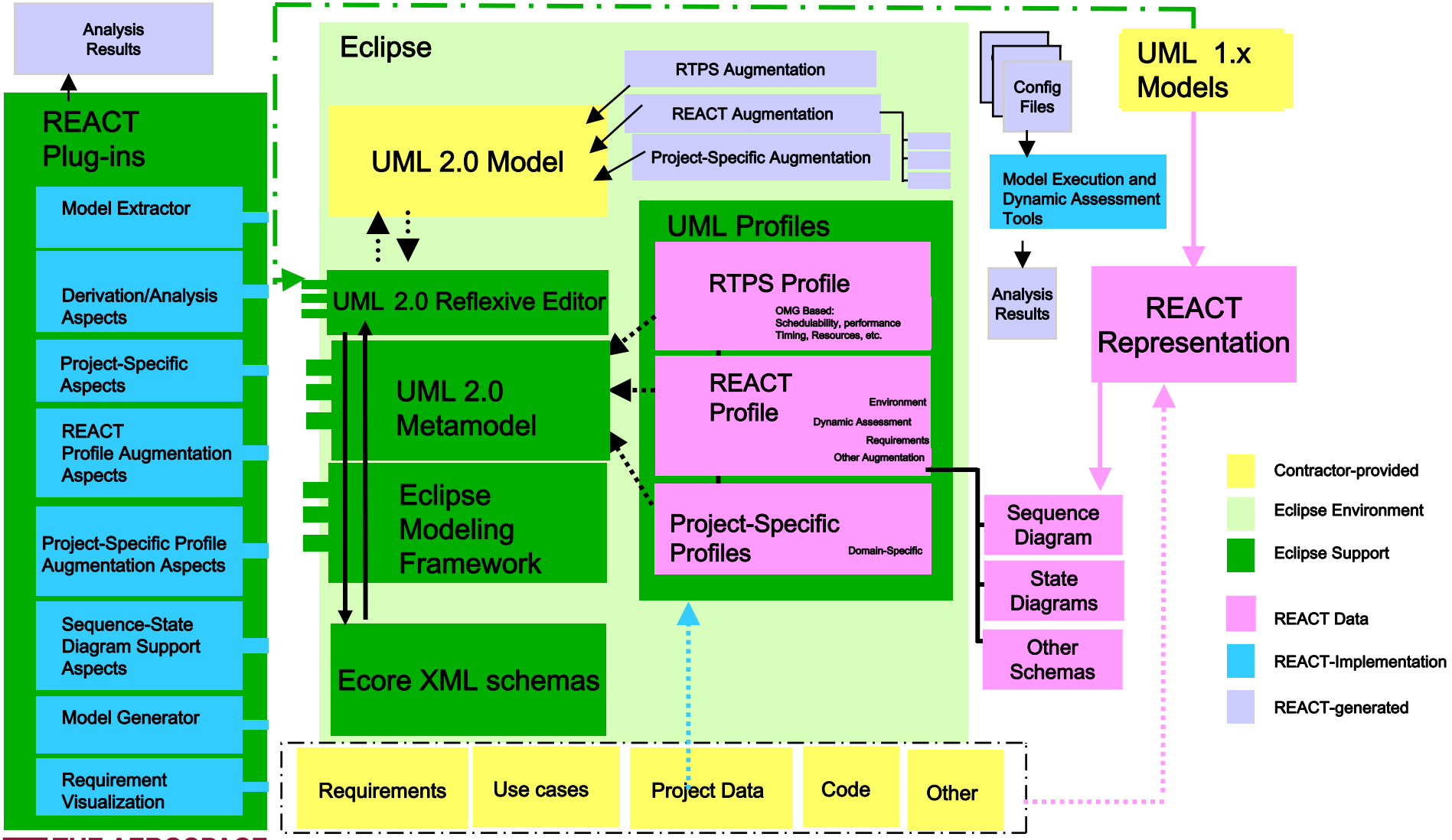
- Periodic assessment more costly
- Concern management more general than feature management
  - Not everything is preplanned
- Aspect mining helpful in investigating potential problems
  - Multiple inter-related representation spaces
  - A code-centric viewpoint was inadequate for addressing unexpected change impacts, subsystem test dependencies, schedule/resource dependencies
- UML2 profiles/XML schemas as a framework for architectural augmentation
  - Requirements evolution
  - Constraint enforcement
  - Parameterizing real-time embedded systems for analysis

References

- Aspect-Oriented Architectural Assessment
  - Schmidt, P., Milstein, J., Alvarado, S., “An Analysis of Aerospace Software Architectures Using Aspect-Oriented Programming Principles,” ATR-2004(8343)-1, 28 May 2004.
  - Schmidt, P. “Representing and Evaluating Software-Intensive Architectures that Evolve,” ATR-2004(8343)-3, 30 Sep 2004.
- Related UML Approaches
  - Selonen,P., Xu, J. “Validating UML Models against Architectural Profiles,” ESEC/FSE 2003, September 1-5, 2003, Helsinki, Finland
  - Hakala,M.,et al. “Annotating Reusable Software Architectures with Specialization Patterns,” http://practice.cs.tut.fi
- Scenario-Based Approaches
  - Bass, L., Clements, P., Kazman R., *Software Architecture in Practice*, Addison Wesley, 1998.
  - Bosch J., *Design and use of Software Architectures: Adopting and evolving a product-line approach*, Addison Wesley, 2000.
  - Clements, P., Kazman, R., Klein, M. *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley, 2001.

Backup Charts

REACT Implementation Elements



THE AEROSPACE CORPORATION