Model Clashes and MBASE

Model-Based (System) Architecting and Software Engineering (MBASE)

Barry Boehm, USC
USC-CSE Annual Research Review
February 9, 1999
(boehm@sunset.usc.edu)
(http://sunset.usc.edu)

Outline

- Model Clash Taxonomy and examples
- Reconciling model clashes
  - Model clash stereotypes
  - MBASE integration framework
  - MBASE process framework
- MBASE results to date
"No scene from prehistory is quite so vivid as that of the mortal struggles of great beasts in the tar pits.

Large system programming has over the past decade been such a tar pit, and many great and powerful beasts have thrashed violently in it."

Fred Brooks, 1975

"Everyone seems to have been surprised by the stickiness of the problem, and it is hard to discern the nature of it.

But we must try to understand it if we are to solve it."

Fred Brooks, 1975
Understanding the Tar Pit: Model Clashes

- Model (Webster): A description or analogy used to help visualize or analyze something; a pattern of something to be made
  - Includes product models, process models, property models, success models
- Model Clash: An incompatibility among the underlying assumptions of a set of models
  - Produces conflicts, confusion, mistrust, frustration, rework, throwaway systems
- Model Integration: Choosing and/or reengineering models to reconcile their underlying assumptions.

Examples of Model Clashes

- Product Model Clashes: structure clashes, traceability clashes, architectural style clashes
- COTS-driven product and Waterfall process
- Risk-based process and spec-based progress payments
- Design-to-cost process and tightly-coupled architecture
- Incremental process and Rayleigh-curve staffing model
- Evolutionary development without life-cycle architecture
- Golden Rule and stakeholder win-win
- Spec-based process and IKIWISI success model
  - I'll know it when I see it
The Golden Rule as Software Success Model

- Do unto others
- As you would have others do unto you
- Build computer systems to serve users and operators
- Assuming users and operators like to write programs, and know computer science
- Computer science world (Compilers, OS, etc.)
  - Users love powerful, obscure, UNIX-like commands
- Applications world
  - Users are pilots, doctors, tellers: Keep it simple
- Better to use Modified Golden Rule
  - Do unto others as you would have others do unto you
    - If you were like them
Clashes Among MBASE Models

<table>
<thead>
<tr>
<th>Product Model</th>
<th>Process Model</th>
<th>Property Model</th>
<th>Success Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure clash</td>
<td>COTS-driven product vs. Waterfall (requirements-driven) process</td>
<td>Interdependent multiprocessor product vs. linear performance scalability model</td>
<td>4GL-based product vs. low development cost and performance scalability</td>
</tr>
<tr>
<td>Traceability clash</td>
<td>Multi-increment development process vs. single-increment support tools</td>
<td>Evolutionary development process vs. Rayleigh-curve cost model</td>
<td>Waterfall process model vs. &quot;I'll know it when I see it&quot; (IKIWISI) prototyping success model</td>
</tr>
<tr>
<td>Architecture style clash</td>
<td>Minimize cost and schedule vs. maximize quality (Quality is free)</td>
<td>Fixed-price contract vs. easy-to-change, volatile requirements</td>
<td>Golden Rule vs. stakeholder win-win</td>
</tr>
</tbody>
</table>

Where do Models (and Clashes) Come From?

- Childhood training
  - Golden Rule, easiest - first
- Past experience
  - Waterfall, Rayleigh curve
- Exaggerating for effect
  - Quality is free, COTS marketing
- Government/Corporate policy
  - Use waterfall, use COTS, use Ada, use 4GL's, Cost as Independent Variable
Usual Model Clash Result

- Initial project honeymoon
- Increasing feeling of tar-pit viscosity
- Attempt to apply surface remedies
  - Baseline the requirements
  - Fire the manager
  - Add more QA people
  - Buy some tools
  - Impose more standards
- Viscosity, confusion, mistrust, frustration, rework persists
  - Often throwaway systems

Outline

- Model Clash Taxonomy and examples
- Reconciling model clashes
  - Model clash stereotypes
  - MBASE integration framework
  - MBASE process framework
- MBASE results to date
Model Clash Stereotypes

• Several stereotypes characterized
  - Government acquisition
  - Entrepreneur
  - Internal services

• Stereotypes include
  - Usual stakeholders
  - Usual stakeholder success models
  - Likely clashes among success models
  - Documented examples of disaster projects and likely clashes they didn’t resolve
    - Largely from Flowers, Glass books

2/9/99 ©USC-CSE

Government Acquisition Stereotype:
Usual Stakeholders

• Users
• Acquirers
• Sponsors
• Maintainers
• Developers
Govt. Stakeholders and Success Models - I

• **Core Users (who?):** Voice in acquisition; feature set scope and priorities; applications compatibility and controllability; low entry barrier; -ilities; rapid mission change accommodation; early availability

• **Acquirers:** Mission cost-effectiveness; Gov't standards compliance; development visibility and control; limited budget and schedule; protest-avoidance; fixed requirements

2/9/99

Govt. Stakeholders and Success Models - II

• **Sponsors:** Political correctness; ambitious budget & schedule; Govt. Standards compliance; protest avoidance

• **Maintainers:** Voice in acquisition; evolution control; -ilities; ease of legacy software migration

• **Developers:** Win source-selection competition; reuse of existing assets; reuse of developed assets; stable requirements; unconstrained development process

2/9/99
Potential Model Clashes: Govt. Contracting - I

1. Product Rqts. Flexibility (User) / Process, Rqts. Stability (Developer)
2. Extensive Product Rqts. (User) / Tight Schedule, Budget (User, Acquirer, Sponsor) / Win Competition (Developer)
3. Applications Compatibility (User, Maintainer) / Reuse of Existing or Developed Assets (Developer)
4. Reuse of Assets (Developer) / Utilities (User) / Tight Schedule, Budget (Acquirer, Sponsor)

Potential Model Clashes: Govt. Contracting - II

5. Development Visibility and Control (Acquirer) / Unconstrained Process (Developer)
6. Tight Schedule, Budget (Acquirer, Sponsor) / Fixed Rqts. (Acquirer) / Voice in Acquisition (User, Maintainer)
7. Political Correctness (Sponsor) / Mission Cost-Effectiveness (Acquirer)
8. Govt. Standards Compliance (Sponsor, Acquirer) / Reuse of Assets (developer) / Standards Incompatibility
9. Protest Avoidance (Acquirer, Sponsor) / Unrealistic Bid to Win Competition (Developer)
Govt. Contracting Model Clashes:
London Ambulance - I


2. Extensive Product Rqts. (User): ambitious highly-automated system / Tight Schedule, Budget (Acquirer, Sponsor): $2.25M, 7 months / Unrealistic Bid (Developer): $1.5M, 7 months (cancelled after 18 months)

4. Reuse of Assets (Developer): Visual Basic for rapid devel. / -ilities (User): Inadequate V.B. performance, scalability, dependability

5. Devel. Visibility & Control (acquirer): weak supervision / Unconstrained Process (Developer): design for nominal case only; perform own quality assurance


Govt. Contracting Model Clashes: Denver Baggage - I


3. Applications Compatibility (User, Maintainer): Airline, airport config. control of facilities / Reuse of Assets (Developer): try to reuse United system for full airport

4. Reuse of Assets (Developer): United baggage system / -ilities (User): safety, dependability / Tight Budget & Schedule: no time to fully test United software: many failures

7. Political Correctness (Sponsor): Issue many contracts to Denver-area firms / Mission Cost-Effectiveness (Acquirer): Too many interfaces, talent shortages, lack of contractor control


Govt. Contracting Model Clashes: Denver Baggage - II
MBASE Integration Framework

Success Models
Win-Win; IKIWISI; Business-Case; Mission Models;...

Process Models
- Life Cycle
  - Waterfall;
  - Evolutionary;
  - Incremental;
  - WW Spiral
- Anchor Points
- Risk Mgmt.
- Activities
  - CMM KPA's

Entry/Exit Criteria
Product Development
Milestone Content;
& Evolution Process
Planning & Control
Evaluation & Analysis

V&V Criteria

Product Models
- Domain
- Artifacts
  - Rqts.
  - Arch.
  - Code
  - Doc'n
- Packaging
  - Embedded
  - Shrink Wrap
  - Turn Key
- Product Line

Property Models
Cost & Schedule; Performance; Assurance; Usability;...

Product Line Domain Scope a Function of ROI, Scope of Empowered PL Manager

Return on Investment (ROI)

Breadth of Domain

too few instances to generate payoff
too general to be competitive

©USC-CSE
## MBASE Conceptual Framework

![MBASE Conceptual Framework Diagram](image)

### Success Models Drive Other Model Choices

<table>
<thead>
<tr>
<th>Success Model</th>
<th>Demo agent-based E-commerce system at COMDEX in 9 months</th>
<th>Safe air traffic control system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Stakeholders</td>
<td>Entrepreneurs, venture capitalists, customers</td>
<td>Controllers, Govt. agencies, developers</td>
</tr>
<tr>
<td>Key Property Models</td>
<td>Schedule estimation</td>
<td>Safety models</td>
</tr>
<tr>
<td>Process Model</td>
<td>Design-to-schedule</td>
<td>Initial spiral to risk-manage COTS, etc.; Final waterfall to verify safety provisions</td>
</tr>
<tr>
<td>Product Model</td>
<td>Domain constrained by schedule; architected for ease in dropping features to meet schedule</td>
<td>Architected for fault tolerance, ease of safety verification</td>
</tr>
</tbody>
</table>

2/9/99 ©USC-CSE
### Elements of Critical Front End Milestones

<table>
<thead>
<tr>
<th>Milestone Element</th>
<th>Life Cycle Objectives (LCO)</th>
<th>Life Cycle Architecture (LCA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of Operational Concept</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Prototype(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition of System Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition of System Architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition of Life-Cycle Plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feasibility Rationale</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### System Prototype(s)
- Define the prototype architecture and its evolution.
- Identify key features and functions.
- Define the prototype's operational environment.
- Identify key technologies.

#### Definition of System Requirements
- Define the requirements for the system.
- Identify the system's operational environment.
- Define the system's operational environment.
- Identify the system's technical requirements.

#### Definition of System Architecture
- Define the system's architecture.
- Identify the system's technical requirements.
- Define the system's operational environment.
- Identify the system's technical requirements.

#### Definition of Life-Cycle Plan
- Define the life-cycle plan.
- Identify the system's technical requirements.
- Define the system's operational environment.
- Identify the system's technical requirements.

#### Feasibility Rationale
- Define the feasibility rationale.
- Identify the system's technical requirements.
- Define the system's operational environment.
- Identify the system's technical requirements.

---

### Engineering Information Set Evolution

#### Engineering Stage

<table>
<thead>
<tr>
<th>Iterations</th>
<th>LCO</th>
<th>Architecture</th>
<th>Iterations</th>
<th>LCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>D</td>
<td>I</td>
<td>E</td>
<td>M</td>
</tr>
<tr>
<td>L</td>
<td>C</td>
<td>O</td>
<td>T</td>
<td>E</td>
</tr>
</tbody>
</table>

#### Manufacturing Stage

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Usable</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>O</td>
<td>T</td>
</tr>
</tbody>
</table>

---

2/9/99

*Univen Ny of Sotahom Center for Software Engineering*
Architecture in a Project's Life Cycle

It encompasses the requirements, architecture and high level design phases of the typical waterfall diagram. It also continues throughout the life of the project (someone continues to wear the architect's hat).

Architecture Phase

Iterative process until consensus is reached. Carries through the life of the project.

MBASE Model Integration: LCO Stage

Determines criteria for new cycle.

Validates readiness of the LCO Model package.
MBASE Results to Date

- Successfully used on over 50 digital library projects
  - Extensively instrumented and analyzed
  - Guidelines improved annually
- Precursor successfully used on large systems
  - e.g., TRW CCFDS-R
- Stereotypes help diagnose early problem situations
- Concepts being increasingly adopted
  - Rational, Xerox, FAA, AF C2ISR Center, others