Architectures for Software Production

Walt Scacchi
ATRIUM Laboratory and
Center for Software Engineering
University of Southern California
213-740-4782, 213-740-8494 (fax).
Email: Scacchi@gilligan.usc.edu
http://www.usc.edu/dept/ATRIUM

This presentation can be found on the WWW at the URL:

http://www.usc.edu/dept/ATRIUM/Architectures.html

Copyright © 1997 Walt Scacchi, The University of Southern California. All RIGHTS RESERVED.

Background

Architectures for Software Production Integrate

- Product
- Process
- Organization Roles (People)
- Development Tools and Assets
- Information Infrastructure
- Organizational Suprastructure

Overview

- Software Production Goals
- Styles and Heterogeneity
- Languages and Support Environments
- Components, Connectors, Configurations
- Architecture Evolution Development and Analysis
- Understanding Architectural Co-Evolution
- Conclusions
Software Production Goals and Strategies

- Creation of (sustainable) profits
- Perform interesting technical work, etc.
- Produce software in timely, cost-effective, and high-quality manner
- Advance technology and understanding through R&D
- Enable evolutionary development of platforms or product lines
- Create differentiated products that enable competitive market advantage
- Alternative goals involve optimality trade-offs
- Strategic alignment of goals is the purpose of architectures for software production

Architectural Styles for Software Production

- Production architectures have characteristic styles
- Styles characterize coarse-grain system features and arrangements
- Styles may be hierarchically cascaded or composed
- Styles are precursors to design patterns and development frameworks
- Complex systems exhibit heterogeneous or hybrid styles.
- Software products, processes, organizations, tools, and infrastructures all have styles.
- Software production may exhibit multiple, heterogeneous styles
- Product styles dominate software architecture discourse
- Conflicting styles cause mis-matches, mis-understandings, and systemic inefficiencies
- Disproportionate focus on style of any production architecture component can lead to sub-optimal software production
- Current ontology of software architectural styles tends not to scale well to:
  - Very large or complex systems
  - Software production
- We need to (re)think software production architecture styles and ontology in terms of strategic alignment.

Architectural Languages

- Current software architecture languages stress product features
- Architectural languages need to also address process, tools, organizational roles, infrastructure, and their features.
- Use of architectural languages focuses on quality virtues (i.e., consistency, completeness, correctness, traceability)
- Need architectural languages to specify software production strategies
- Need to extend use of architectural languages to assess or optimize strategic alignment of software production.
Components, Connectors, Configurations

- Software products, processes, organization roles, tools, and infrastructure all have components, interfaces, connectors, and configurations.
- The configuration of components whose interfaces are interconnected with components define a software architecture.
- Components, connectors ("relational functions"), and configurations all have types, attributes, relations, and versions.
- Architectural configurations appear at the different levels of abstraction:
  - **Conceptual**: the "one page diagram" level (i.e., a rendering).
  - **Logical**: a structural arrangement of modules that may reveal overall modularity, data/control flow, and sub-system styles.
  - **Physical**: a layout of actual code modules, their interconnections, and connectors (can identify >100K components for very large systems).
- Different levels of architectural abstraction reveal different kinds of insights about software production and strategic alignment.

Architecture Evolution, Development, and Analysis

- Inappropriate combination of architecture components or connectors causes architectural mis-matches.
- Architectures drift over time due to inappropriately combined or inadequately analyzed evolutionary changes.
- Embedded architectures thus become difficult and costly to dislodge and displace.
- Production architectures that drift can engender counter-implementation activities that exacerbate their replacement or work-arounds.
- Procrustean legacy architectures emerge due to drift.
- Software production architectures can be specified, prototyped, simulated, transformed, compiled, executed, or automatically generated.
- Architectures can be analyzed for their consistency, completeness, correctness, traceability, optimization, and execution performance.
- Systematic development and analysis can mitigate mis-match and drift in software production architectures.
Understanding Architectural Co-Evolution

- Software products, processes, organizational roles, tools, and infrastructure are inter-related
  - *Generic Example:* People (organizational roles) perform processes using tools that create, access, and update products over an infrastructure.
- Software production architecture components thus can co-evolve.
- Software production architectures represent an ecology of co-evolutionary dynamics and interactions
- Software production architectures can evolve, drift, and embed around their component architectures

Conclusions

- The purpose of architectures for software production is the strategic alignment of goals.
- We need to (re)think software production architecture styles and ontology in terms of strategic alignment (e.g., strategic styles and style strategies).
- Need architectural languages to specify software production strategies
- Need to extend use of architectural languages to assess or optimize strategic alignment of software production.
- Different levels of architectural abstraction reveal different kinds of insights about software production and strategic alignment.
  - Need tools and techniques for surfacing, visualizing, and understanding these insights
- Need tools to detect or anticipate software production architectural drift, and its consequences beforehand.

This presentation was last updated by Walt Scacchi on 3 March 1997.