State of the Art: Software Architecture

Dewayne E. Perry

Bell Laboratories
Room 2B-431
600 Mountain Ave
Murray Hill NJ 07974
dep@research.bell-labs.com
www.bell-labs.com/~dep/

May 1997
ICSE ’97

Apologia

• This SOTA talk is a mixture of
  – State of the practice
  – State of the research
  – State of the problem

May 1997
ICSE ’97

Overview

4 Rationale for a separate discipline
• Models of software architecture and style
• Architectural specifications and their uses
• Codification of architectures, styles and elements
• Architectural processes
• Where next?

May 1997
ICSE ’97

Rationale

• The architecture is there whether we make it explicit or not
• If it is implicit, then we have no way of
  – controlling it,
  – analyzing it,
  – reasoning about it, or
  – evolving it.

May 1997
ICSE ’97

Rationale

• If it is implicit, we may not be able to
  – identify it,
  – understand it,
  – conform to it, or
  – maintain it.
• Basic Architectural problems
  – architectural mismatch
  – architectural drift
  – architectural erosion

May 1997
ICSE ’97

Rationale

• Benefits of an explicit architecture
• Establishes the structure for satisfying system drivers
  – User/Market Requirements
  – Domain requirements
  – Business constraints
  – Product-line constraints
  – Project constraints

May 1997
ICSE ’97
Rationale

- Defines the important structural aspects
  - The load-bearing walls,
  - The components, their properties and relationships,
  - The styles of initialization, fault recovery, reliability, etc

- Provides a structural framework for
  - System development,
  - System evolution,
  - Component design and implementation,
  - Asset generation and use/reuse, and
  - System composition

Rationale

- Differences between Architecture and Design
  - Architecture is concerned about higher level issues
  - components vs procedures
  - interactions among components vs interfaces
  - constraints on components and interactions vs algorithms, procedures and types

- Architecture is concerned with a different set of structural issues
  - Large-grained composition vs procedural composition
  - Component interactions (protocols) vs procedural/task interactions (pc, rpc, msgs, etc)
  - Information content vs data types and representations

Overview

- Rationale for a separate discipline

4 Models of software architecture and style

- Architectural specifications and their uses
- Codification of architectures, styles and elements
- Architectural processes
- Where next?

Models of SW Architecture

- Perry & Wolf 89/92 model of SW Architecture
- SW Architecture = ( Elements, Form, Rationale )
- Elements : process, data and connecting
- Form is the set of properties of, and relationships among, the elements
- Rationale is the justification for the elements and form
Models of SW Architecture

- State of current models
  - Pretty much agree about process, data and connecting elements (often referred to as components and connectors) as first class entities
  - Models differ primarily with respect to Form
  - Few models pay attention to rationale
  - Styles tend to focus on element and form restrictions

Models of SW Architecture

- A characterization of existing models
  - Configuration
  - Type
  - Pattern
  - Property

Summarization for each model includes

- Characterization
- Approach to style

Examples of this Approach

- Kramer & Magee - dynamic structures
- Le Metayer - graph grammars as styles
- Configuration important in other models

Model - Configuration

- Characterization
  - Basic box and lines approach
  - Components may be processes, subsystems, etc
  - Connections are defined by Provides/Requires clauses

- Approach to Style
  - Tend not to be interested in styles
  - Except in the context of dynamic arch’s

Examples of this Approach

- Most informal descriptions
- Kramer & Magee - dynamic structures
- Le Metayer - graph grammars as styles

Model - Type

- Characterization
  - Typically, an historical approach
  - Look for types and classes of architectural objects
  - Often organized hierarchically

- Approach to Style
  - Emphasis on the basic classes or types of components and connectors
    - Perhaps, a slight more emphasis on connectors
    - Eg, pipes and filters; blackboard architecture

Examples of this Approach

- Shaw, et al
- Hudak
Model - Pattern

• Characterization
  – Emphasis on patterns of interactions
  – Tendency to focus on connections with components as endpoints

• Approach to Style
  – Architectural instances are specializations of styles

• Examples of this Approach
  – Garlan et al (Wright, etc)
  – Invarardi and Wolf et al (use of CHAM - transformation patterns)
  – Luckham et al (Event patterns)
  – Kramer and Magee (Patterns of interactions)
  – Taylor et al (C2 style)
  – Gamma et al, Siemens (OO patterns)

Model - Property

• Characterization
  – Properties of (or constraints on) data, process and connecting elements
  – Relationships among data, process and connecting elements

• Approach to Style
  – Selection of some critical elements
  – Selection of some properties and relationships
  – Constraints on properties and relationships

• Examples of this Approach
  – Perry and Wolf
  – Moriconi and Qian

Overview

• Rationale for a separate discipline
• Models of software architecture and style
  4 Architectural specifications and their uses
• Codification of architectures, styles and elements
• Architectural processes
• Where next?

Architectural Specifications and their Uses

• Prescription vs Description
• Traceability
• Analysis
• Visualization and simulation
• Configuration/Generation
Architectural Specifications

- Prescription
  - Emphasis on intent, critical aspects
  - Tendency towards minimality/incompleteness
  - Problem domain emphasis
  - Tendency towards high level constraints
- Description
  - Emphasis on what exists
  - Tendency towards completeness
  - Implementation domain emphasis
  - Tendency towards detailed descriptions

Architectural Specifications - Traceability

- Rationale is link between architecture and its drivers
  - Non-functionally induced structure
  - Functionally induced structure
- Mapping to design/implement components

Architectural Specifications - Analysis

- Level of analysis depends on the underlying model and the expressiveness of the specification language
  - Configuration: standard build
  - Type: compiler technology
  - Pattern: model checking and simulation
  - Property: depends on
    - expressibility
    - decidability

- Typical kinds of Analyses
  - Consistency and Completeness
    - configuration completeness
    - configuration consistency
    - component - connector consistency

- Other functional properties
  - safety properties
  - mismatch detection (Irvaradi & Wolf, et al)
  - satisfaction of component by subarchitecture (Moriconi et al)

- Non-functional properties, for example
  - performance
  - reliability
  - Style conformance

Architectural Specifications - Visualization & Simulation

- Graphical versions of text
- Graphical representation of analyses (Kramer & Magee)
  - Full patterns of interactions
  - Minimization of interactions
- Simulation of event patterns (Luckham et al)
- Visualization and simulation of architectural intent
  - Instrumented connectors (Balzer et al)
Overview
• Rationale for a separate discipline
• Models of software architecture and style
• Architectural specifications and their uses
  4 Codification of architectures, styles and elements
• Architectural processes
• Where next?

Codification
• Implementation components
• Type approach
• Property approach
• In general, still a long way to go

Codification - Implementation Components
• Basic Platforms
  – Common components: GUIs, object mgmt, etc
  – Domain-specific: application-specific platforms
    • first step towards a product line architecture
• Shared Assets
  – Motivation: cost, interval leverage
  – first step towards domain specialization
• Serves as basis for architectural generation

Codification - Type Approach
• Classified existing common components and connectors
• Tendency:
  – Functional classification
  – Solution domain
• Codified styles: restriction of component and connector types
  – For example, pipes and filters
Codification - Property Approach (Batory)

- Domain-specific architectural assets
  - Components appropriate to the domain
  - Components defined by properties
- Consistent architectural instance created by
  - Component composition on the basis of desired properties
  - Propagating and satisfying the desired properties (ala Perry’s Inscape)

Codification - Long Way To Go

- Need non-functional properties
- Understanding of interaction between functional and non-functional properties
- Codification in problem domain
  - Domain-specific templates
  - Applicability of codified solution domain components to problem domain components

Overview

- Rationale for a separate discipline
- Models of software architecture and style
- Architectural specifications and their uses
- Codification of architectures, styles and elements

4 Architectural processes
- Where next?

Architecture Processes

- Architecture Recovery
- Domain Specific
- Product Line
- Architecture Evolution

Architecture Recovery Process

- The models are suggestive of a recovery process
  - create a configuration model
  - determine the types of the components and connectors
  - determine the patterns of interactions among the components
  - abstract the properties of and relationships among the components and connectors

Domain Specific

- Problem vs implementation focus
- Create a business domain model
  - different from design level domain analysis (tends to be in the implementation domain)
  - Eg, messaging in terms of email, voicemail, fax, multi-media, etc - not GUI and DBs
- Populate architectural model with domain elements and appropriate interactions
Product Line - Basic Aspects

- Begin with product instances
  - legacy based
  - use architecture recovery processes
- Focus on appropriate business domain
  - use domain specific architectural processes
  - map from recovered to domain architecture
- Abstract/Generalize to Product Line Architecture

Product Line - Overview

- Product Line Reference Architecture
- Product Line Processes
- Products
- Asset Base

Product Line - Issues

- Product Line Reference Architecture
- Product Line Processes
- Asset Base
- Supporting Technology
- Organizational Issues

Product Line - Ref Architecture

- Domain-specific prescription or description
- Parameterized architectural components
- Refinement into sub-architectures
- Style descriptions for
  - critical architectural aspects
  - orthogonal aspects - eg, initialization, fault recovery, etc

Product Line Processes

- Create/evolve the reference architecture
- Create/evolve architectural instances
  - instantiate
  - provision
  - configure/generate
- Create/evolve asset base
  - shared components
  - specialized components
  - Use asset base for architectural instance implementation

Asset Base

- Design component descriptions
  - common interfaces
  - common implementations
  - product-specific implementations
- Various supporting platforms
- Product specific components
Supporting Technology

- Architecture
  - Analysis - sufficiency, satisfaction
  - Instantiating, provisioning, customization
  - Generation/configuration
- Design/Implementation
  - Architecture satisfaction analysis
  - Component composition/analysis
  - Connector optimization
  - Runtime generation

Organizational Considerations

- Architecture/Asset base
  - Across product lines
  - Product line specific
  - Product specific
- Supporting technology
  - Global to the company
- Processes - support multiple product lines

Architectural Evolution

- From prototype to embedded system
  - Transformations (Tracz, et al)
- Importance of connectors - significant for
  - Evolving non-functional properties
  - Improving problem solution (Balzer et al, instrumented connectors)

Architectural Evolution

- Dynamic evolution
  - Allowed dynamic changes
    - Creation/destruction of components and connectors (Kramer & Magee)
    - To respond to dynamic system requirements
  - Appropriate support for
    - Distribution independence
    - Dynamic linking, registration (Taylor et al)

Overview

- Rationale for a separate discipline
- Models of software architecture and style
- Architectural specifications and their uses
- Codification of architectures, styles and elements
- Architectural processes

Where Next?

- Higher level ADLs
- Generic architectures and customization
- Increased product line architecture support
- Increased codification in both problem and solution domains
- Standardized domain-specific architectures
- Increased generation using codified solution domain

Architectural Evolution
Post-ICSE Addenda

- Garlan et al’s work is also relevant for the configuration model
- Where Next should have included ‘Multiple Views’
- Apologies to Jeff Magee for mispelling his name