Software Architecture Roadmap

This Talk

Overview

- Software architecture has emerged as an important area of concern for practitioners and researchers.
- Considerable progress has been made in developing an engineering basis for architectural design.
- The changing face of software technology raises some exciting new challenges and opportunities.

Talk Outline

- The role of Software Architecture -- what it's good for.
- Evolution of the field -- forces that have shaped its development.
- Challenges and opportunities -- new forces for change.
Software Architecture Roadmap

The Problem

Requirements

How to bridge the gap between requirements and code?

Code

The Traditional Answer

Requirements

A Miracle Happens!

Code

- Ad hoc
- Requires gurus
- Unpredictable
- Costly
The Role of Software Architecture

- Composition of large-scale components
- System-level abstractions
- Reuse of system-level design idioms

Requirements

Software Architecture

Code

Issues Addressed by an Architectural Design

- Gross decomposition of a system into interacting components
  - Typically hierarchical
  - Using rich abstractions for component interaction (or system "glue")
  - Often using common design idioms/styles
- Emergent system properties
  - Performance, throughput, latencies
  - Reliability, security, fault tolerance, evolvability
- Rationale and assignment of function to components
  - Relates requirements and implementations
- Envelope of allowed change
  - "Load-bearing walls", limits of scalability and adaptation
  - Design idioms and styles
How Should a Software Architecture Help?

- Understanding
  - vocabulary for structure, system constraints
- Reuse
  - of components, and also of patterns for organization
- Construction
  - partial blueprint, compilation/construction instructions
- Evolution
  - allowable envelope of change
- Analysis
  - system-level analysis that exploits structural constraints
- Management
  - evaluation-ready milestone

Antecedents of Software Architecture

1950
- Subroutines
- Separate compilation

1960
- Programming-any-which-way

1970
- NATO SE conference
  - Integrated product lines
  - Software architecture
  - Packages
  - Pipes and filters

1980
- Programming-in-the-small
- Programming-in-the-large
- Information hiding
- Abstract data types
- Programming-in-the-world

1990
- Software development environments
- Inheritance
- Subroutines

2000
- Integrated product lines
- Software architecture
- Packages
- Pipes and filters
Software Architecture Roadmap

Evolution of the Field

1980's
- Informal use of box and line diagrams
- Ad hoc application of arch expertise
- Diverse, uncodified use of architectural patterns and styles

1990's
- Recognition of the value of architects in software development organizations
- Processes that require architectural design reviews and explicit architectural documentation
- Emerging use of product line architectures, commercial architectural standards, component integration frameworks
- Codification of vocabulary, notations & tools for architectural design
- Books/courses on software architecture

Maturity: Progressive Codification Cycle

Ad hoc solutions

Folklore

Improved practice

Codification

New problems

Models & theories

Institute for Software Research, International
Forces for Change: Economic

- Changing build-vs-buy balance
  - Time-to-market pressures often make construction from scratch infeasible
  - Successful component frameworks enable rapid construction from third-party parts (e.g., Visual Basic)
  - Industrial standards encourage uniform packaging and provide standard APIs (e.g., Enterprise JavaBeans)
  - Emerging standards raise level of abstraction, promise a basis for compliance (e.g., HLA)

Forces for Change: Technical

- Pervasive Computing
  - Proliferation of computing devices (e.g., toasters, entertainment systems, cars), many with dedicated embedded processors
  - Heterogeneous capabilities (e.g., toasters vs PDAs vs cell phones vs high-end workstations)
  - Mobile computing (e.g., in cars and airplanes)

- Network-centric Computing
  - Enabled by increasing connectivity of systems and information
  - PCs become front ends to services & information available on the net
  - Tasks performed by coalitions of resources enlisted by the user
  - Coalitions must be reconfigured dynamically as resource mix changes
Example: Network-Centric Computing

- Internet provides rich variety of resources
  - information, calculation, communication, services
  - Autonomous, heterogeneous, interactive
- Inevitably, these will be incorporated in systems
  - Coalitions, not true systems—open-shop, not closed shop development
  - Specifications scanty and less than reliable
  - Resources subject to unannounced change
  - Problems brittleness get worse
- Research opportunities
  - How to identify and validate a resource, then incorporate it
  - How to determine what a coalition does and keep it doing that
  - How to make it possible for nonexperts to recruit coalitions

Research Challenges and Opportunities

- Better principles, notations, tools for integration frameworks
  - not just APIs
- SW architectures that scale with size/variability of the Internet
  - millions of nodes, frequent reconfiguration, many formats/protocols
- Open, dynamic resource coalitions
  - no longer under control of single organization or proprietor
- Techniques for bridging architectural mismatch
  - heterogeneous data formats, component technologies
- Ensuring suitable properties for distributed system configurations
  - even though absolute correctness is not achievable
- Architectures for mobility
  - resource-aware, highly dynamic, multi-device
When architects of several minds
Sketch their systems with boxes and lines,
Their frameworks of objects
Allow all their projects
To share in each others’ designs.