End-User Architecting

David Garlan
Carnegie Mellon University, USA

21 March 2012
17th Monterey Workshop

Research Interests

... related to this workshop
• Self-adaptive systems
  Rainbow: architecture-based run-time adaptation
  Cf., Monterey 16 (Seattle)
• End-user architecting
  This talk
• Cyber-physical Systems
  Multi-view modeling
  Teaser at end of this talk

© David Garlan 2012
In a nutshell...

Many domains require end users to compose functionality to automate tasks, procedures, analyses, etc.

This activity is similar to architecting:
- Requires component composition
- ... within domain-specific styles of construction
- ... supporting quality attributes such as performance, security, ...

The concepts of software architecture can be applied to end-user composition to provide
- Abstractions tailored to the user’s domain
- Analyses that provide feedback and guidance
- Execution support

Success requires a clear understanding of the socio-technical ecosystem

Background

End-User Developers:
- People who create and execute programs in support of their professional goals, but not as their primary job function
- Examples: business analysts, neuroscientists, physicists, intelligence analysts, ...

Assembly of computations by end users:
- One of the main activities of such end users is to compose heterogeneous computational entities.
- Today this requires programming expertise
  - These users spend about 40% of their time doing programming activities. [Howison J., 2011]
Some End-User Composition Domains

**Neuroscience:** Process brain-imaging data, apply statistical analysis, and generate reports.

**Dynamic network analysis:** Process unstructured data about an organization/society/community to generate a social-network, and then create reports about communication patterns, key entities, and future trends.

**e-Science:** Perform scientific experiments using large distributed datasets, including physics, astronomy, chemistry, etc.

**Bioinformatics:** Perform interactive large-scale genome analysis by combining data from independent queries.

---

**Business Process Management:** Compose, analyze, reengineer and execute business processes.

**Smart buildings and homes:** Monitor, analyze & control building automation, including energy & security.

**Personal medicine:** Configure the way personal medical information is processed and analyzed.

**Digital Audio:** Compose virtual audio components to synthesize music.
The Problem

Creating compositions today is difficult for end users:

► Complexity from low-level detail
  › For example, parameters, file systems, execution paths, operating systems, data formats, etc.

► Conceptual mismatch
  › For example, “Remove Image Noise” as opposed to invoking the specific program(s) to perform this function.

► Lack of support for error detection and resolution
  › For example, it is hard to know if a composition will work in advance of executing it, or to determine quality attributes such as performance, security and privacy.

► Lack of support for reuse
  › Compositions cannot be easily shared or tailored to new situations.

Example 1: Brain Imaging

The field of brain imaging is an important emerging area, leading to scientific breakthroughs

► There exist large repositories of brain imaging data
  › For example, the Brain Imaging Network (Portugal)

► There also exist dozens (if not hundreds) of brain image processing tools
  › Image recognition, image alignment, filtering, volumetric analytics, mapping, ...

► Innovative research in this domain requires that scientists compose these tools and apply them to large data sets
  › There exist large consortiums of scientists working on these problems, who share data, tools, and findings
Example 1: Brain Imaging

Problem: Multiple Programs can be used, depending on data source.
Example 1: Brain Imaging

Problem: Abundance of parameters to get results right.

/usr/local/fsl/bin/flirt
-ref standard
-in example_func
-out example_func2standard
-omat example_func2standard.mat
-cost corratio -dof 12
-searchrx -90 90
-searchry -90 90
-searchrz -90 90
-interp trilinear

A large script file that contains program calls

Program (a large number of binaries that perform one or more functions)
Parameters (number of parameters typically ranges from 5 to 25)
Example 2: Socio-Cultural Analysis (SCA)

Understand, analyze, and predict relationships in complex social systems
  ► Human “terrain” in military engagement
  ► Criminal activities in a metropolitan area
  ► Business intelligence
  ► Communication of policy changes in cities

Incorporates many theories, tools, approaches
  ► Text/data mining, natural language understanding
  ► Network analysis theory, statistics, decision support
  ► Simulation, game-theory

Haiti Earthquake 2010

Analysis of humanitarian relief effort after Haiti earthquake
  ► Process public domain news sources
    › Filter out headers, remove noise, normalize concepts
  ► Build and analyze a multi-mode network
    › People, organizations, places, relationships, times
  ► Answer questions
    › What organizations were involved and in what way?
    › When did emphasis shift from rescue to finding fresh water?
    › How did local government, NGO, and foreign government relationships affect distribution of relief?
Using SCA Tools

Typical Service-Oriented Composition

• Workflow containing 4 logical processing steps
  – Represented as a BPEL* orchestration
  – Executed on a SOA platform
• Authors must understand:
  – SOA invocation protocols
  – Parameter assignment mechanisms
  – Error handling
  – Time-out constraints

*BPEL = Business Process Execution Language
Example 3: Intelligence Analysis

Ozone Widget Framework (OWF) is an emerging integration framework

 › Enables rapid assembly of widgets.
 › Widgets are single-purpose web-applications that provide summary views of dynamic information content.
 › Framework is being promoted to create a common widget repository – similar to 'Android Market'.

Ozone dashboard for displaying widgets

Deployment for Ozone Widgets

Widgets need not be hosted on the same server, domain, or technology as OWF.
Publish-Subscribe Integration

OWF provides a framework to support cross-domain communication between widgets through messages and channels

- Widgets can publish and subscribe to channels to communicate messages.
- But these must be programmed in scripts.

What is Needed

End users need a solution that

- Allows them to compose existing tools, services, applications, data, and other compositions
- Without detailed technical expertise
- In a language appropriate for their domain
- Supported by construction and execution tools that allow them to
  - create and run these computations
  - analyze them for relevant behaviors (such as design errors).

INSIGHT: This is similar to Software Architecture!
What is Software Architecture?

- High level of design abstractions and analysis
- Emergent properties focus
- Reuse of design styles, patterns, frameworks

The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software elements, relations among them and properties of both.
The Rise of Software Architecture

Softw...
## Architecture Design Tools

Support for domain-specific architecture development
- Style design, visualization, compilation, ...

Analysis tools
- Component mismatch, performance, reliability, security, ...

Support for multiple views
- Code, run-time, deployment, ...

Linkage to organizational processes
- Documentation, review, evolution, ...

---

## An End-User Architecture Approach

An architectural approach to end-user composition means

- Existing architectural techniques can be used for defining the domain, supporting composition, aiding in understanding trade-offs

Three key elements to the approach
1. An **architecture layer** between the user interface and execution environment supports explicit representation of end-user compositions
2. A **reusable style** that can be specialized for specific domains
3. A **graphical front end** for composition and for analyzing and executing compositions
Architecture Layer

Architectural Layer

Execution Platform

Primitives

Domain-specific architectural styles

Platforms like SOA, SCA etc

Programs, tools or services

Guidance  Analysis  Translation

Architectural Layer

Monitoring  Code Generation  Adaptation

Execution Platform

Primitives

Visual Language

Domain-specific architectural styles

Platforms like SOA, SCA etc

Programs, tools or services
**SCORE** (Simple Compositional ORchestration for End users)

**SCORE Root Style**

- **SCORE Family**
- DataStore
- Service
- Tool
- Library

**SCORE Specialization for Neuroscience**

- NeuroScience Family
  - VolumeData
  - Segmentation
  - Normalize
  - Align
  - Registration
  - Temporal Filtering
  - Spatial Filtering
  - FSL Family
  - flirt
  - bet2
  - fsl
  - mcflirt
  - sum

**SCORE Specialization for Intelligence Analysis**

- SORASCS Services & Tools Family
  - Task
  - Visualizer
  - Procedures
  - Data Service
  - Reporting Service
  - Workflow Service
  - ORA
  - Context
  - AutoMap
  - Third-party Tools

© David Garlan 2012
Specializing SCORE: Intelligence Analysis

Specializing SCORE: Neuroscience

“Missing Alignment before Temporal Filtering”
SORASCS System Organization

SORASCS Implementation

Built on standard, open-source SOA technologies:

► Apache Tomcat web server
  › Provides web-based access to applications and web services

► Apache CXF
  › Provides method for turning existing Java applications into web services

► Apache ODE
  › Provides BPEL execution engine for service orchestrations

► SOAP/WSDL for Web Service communication

Currently more than 120 Services and 10 standalone tools integrated.

In use today by US intelligence community
Learning from End-user Architecting

Usually missing from Architecture Design
Environments – open areas for research
• Component repositories
  • Ways to contribute, find, document, certify, and reuse components
  • Can be difficult when you have hundreds of components
• Mismatch repair
  • Components often do not work together “out of the box”
  • Require ability to detect and repair mismatch
• Packaging and parameterization
  • Encapsulating common structures and patterns
  • Being able to easily instantiate these and combine them
• Pedigree, provenance, credibility
  • Common problems: tracking results, understanding how well one can trust the outputs

© David Garlan 2012

Beyond Architecture: Sustainability

It is not enough to have a good platform, interface, and set of components
To be successful we require sustainability
This, in turn, requires a stable Socio-Technical Ecosystem

A Socio-Technical Ecosystem (STE) represents a complex, self-sustaining system including:
▶ Stakeholders of various types
▶ Incentive systems for different stakeholders to participate in the ecosystem
▶ Appropriate organizational, governmental (legal), economic, social structures
▶ A technical architecture – usually a platform

© David Garlan 2012
Example: Structure of the Mainframe Computer Industry

Example: New Structure Emerges
Modern STEs

Single product-oriented STE
- Develops and sells a software product
- Architecture: single system, traditional architecture

Product-line STE
- Company develops a product line
- Architecture: proprietary framework with extension points

Service-oriented STE
- Many companies develop services
- Architecture: SOA

Platform-oriented STE
- One organization develops/maintains a platform; third parties create extensions
- Architecture: Platform+Plug-ins (Apps)

End-User Architecture STEs

Roles
- Tool/service developers
- Platform developers/maintainers
- End-users
- Governance body

Incentive System
- What motivates each of these roles to do their part?

External Forces
- Government regulation? (e.g., privacy)
- Economic benefits? (e.g., charge for tool use)
- Social constraints? (e.g., how does the community interact?)
Example

We constructed a platform for socio-cognitive analysis, shown earlier

- Tool/service developers: researchers in sociology, anthropology, social networks
- Platform developers/maintainers: our research lab
- End-users: analysts
- Governance body: our research lab

Technically a great success!

Problems

- Missing incentive system
- Government regulation made widespread use impossible because of certification rules.
- Social constraints made it difficult to get researchers to provide their tools to us.

Conclusions

- End users can create complex systems using architectural abstractions
  › Matched to domain and computational intuition
  › Analyzed through architectural analysis
  › Automatically translated into low-level code and interactively executed
- A framework that promotes such a design can help
  › Reuse standard architectural mechanisms and tools such as architecture styles and analyses
  › Provide platforms for integration and execution
More conclusions

► Experience with end-user architecting suggests some open research issues for architecture tool developers
► Long-term success requires a sustainable socio-technical ecosystem
  › Incentive systems
  › Governance bodies
  › Regulatory and legal climate
  › Organizational and social structures

Acknowledgements

This is joint work* with
• Vishal Dwivedi, Bradley Schmerl, Kathleen Carley
  • Carnegie Mellon University, USA
• Perla Velasco-Elizondo
  • Centre for Mathematical Research (CIMAT), Mexico
• José Maria Fernandes
  • IEETA/DETI, University of Aveiro, Portugal

*... supported by grants from the US Dept. of Defense, the National Science Foundation, and the Portuguese government
References


Some End-User Composition Tools

© David Garlan 2012