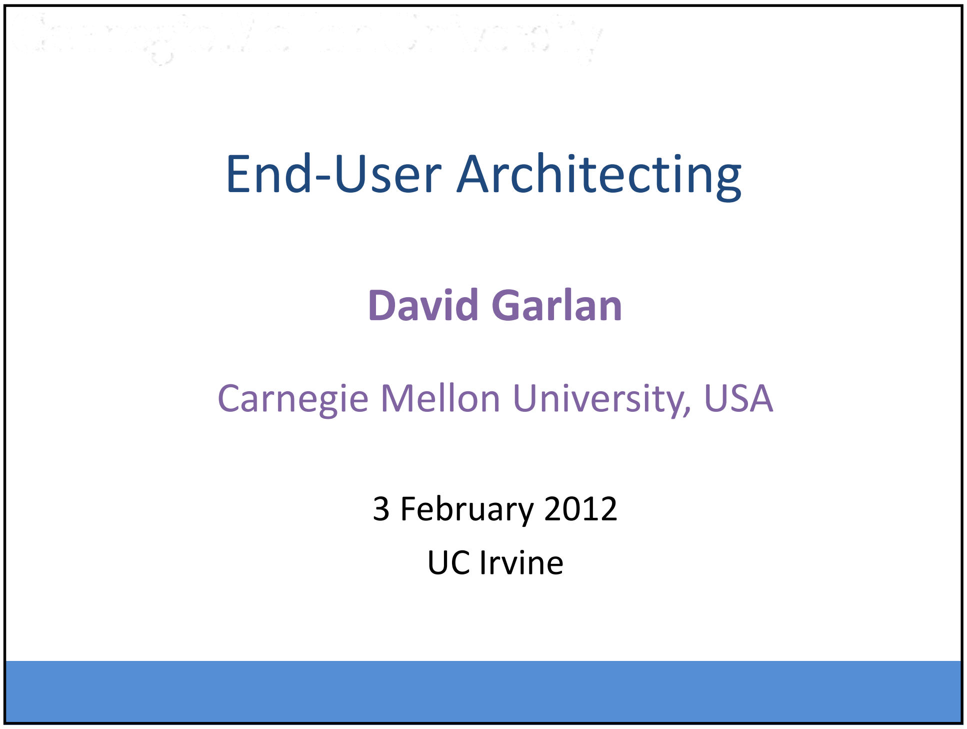
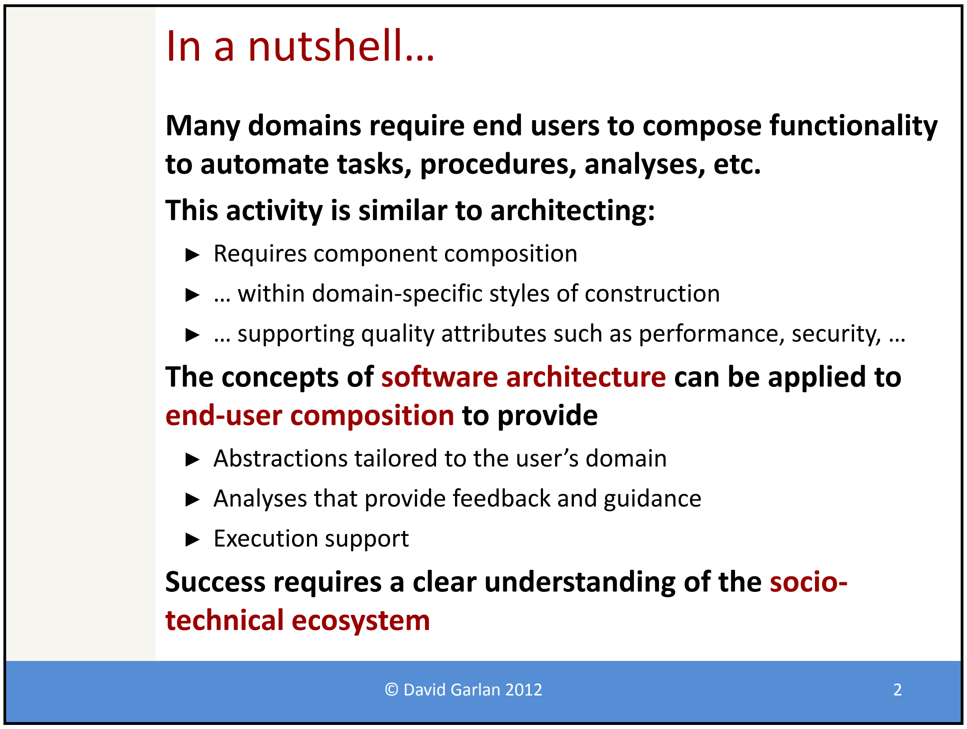


End-User Architecting

David Garlan

Carnegie Mellon University, USA

3 February 2012
UC Irvine



In a nutshell…

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

This activity is similar to architecting:

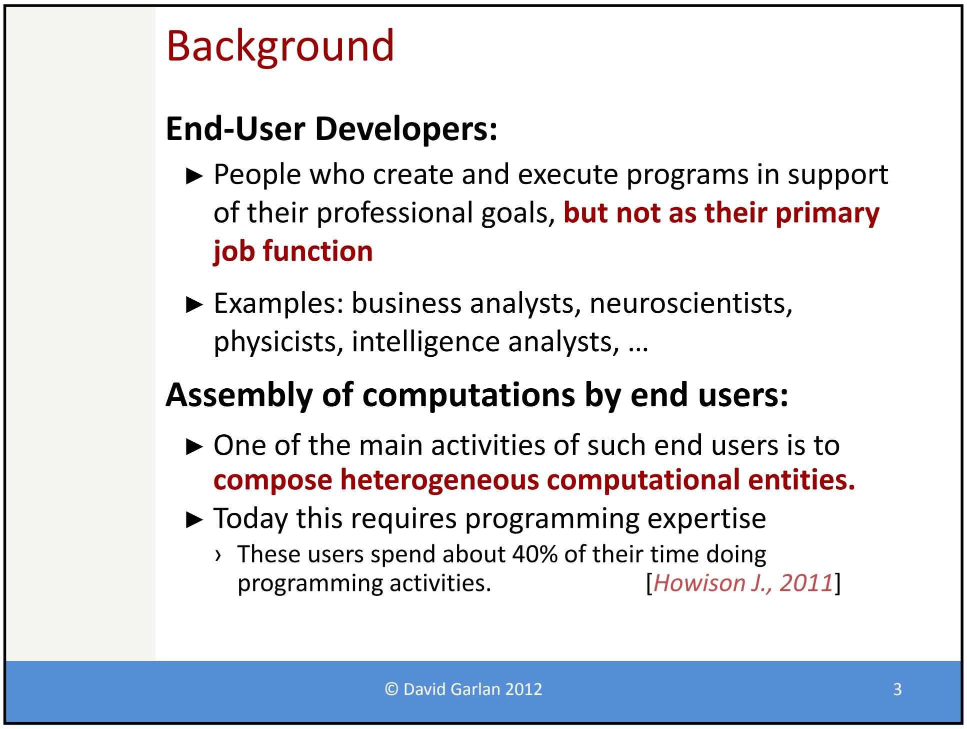
- Requires component composition
- … within domain-specific styles of construction
- … supporting quality attributes such as performance, security, …

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

2



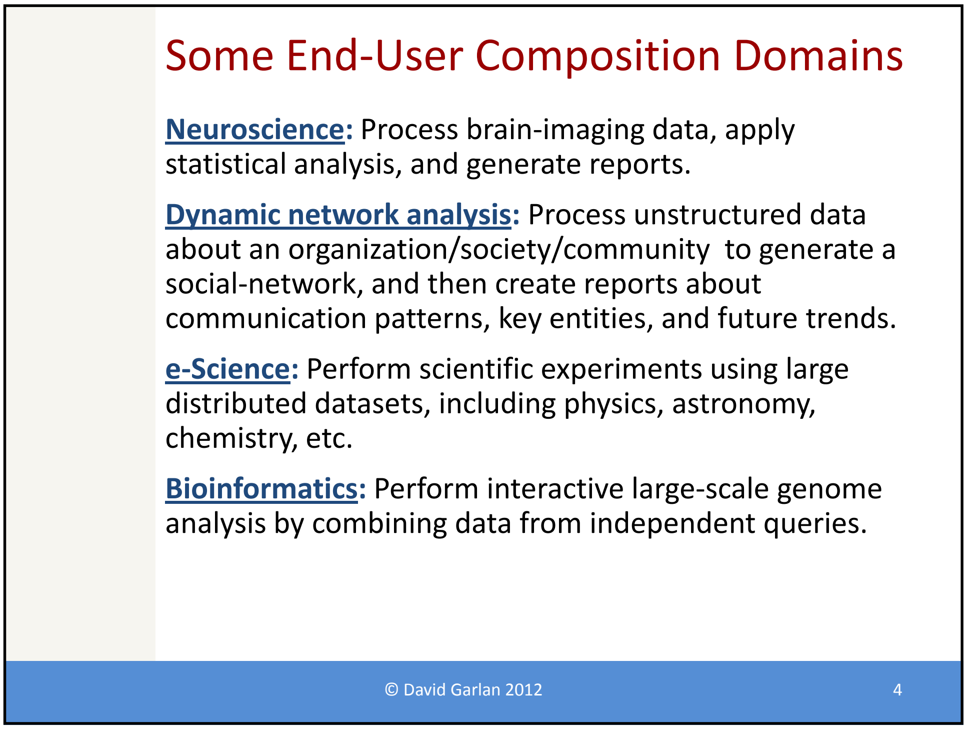
Background

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*]



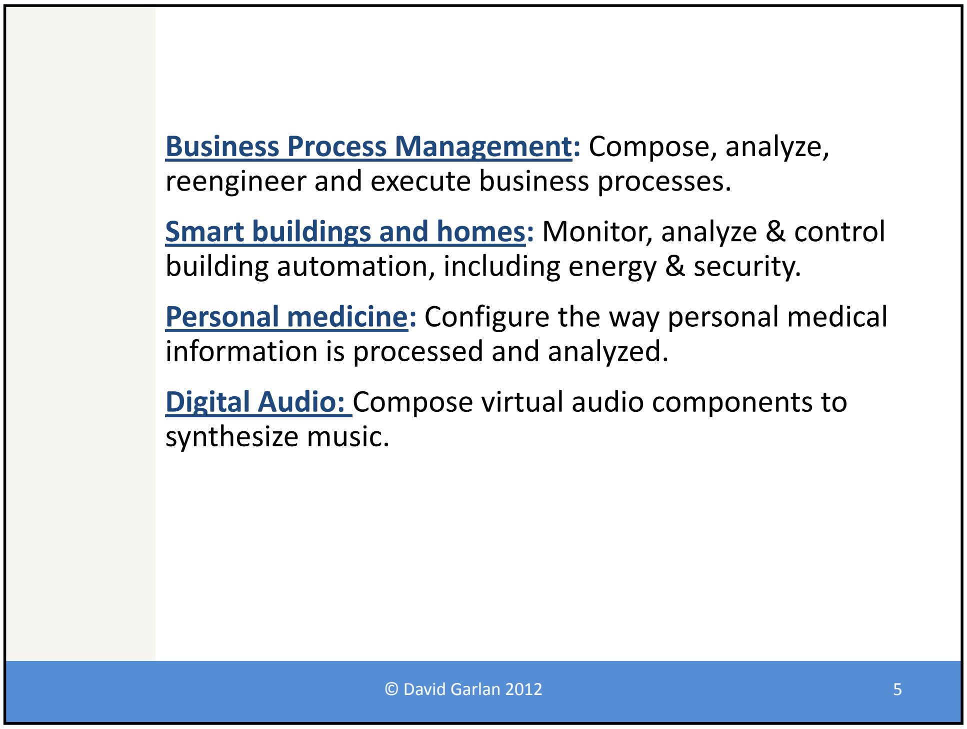
Some End-User Composition Domains

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.

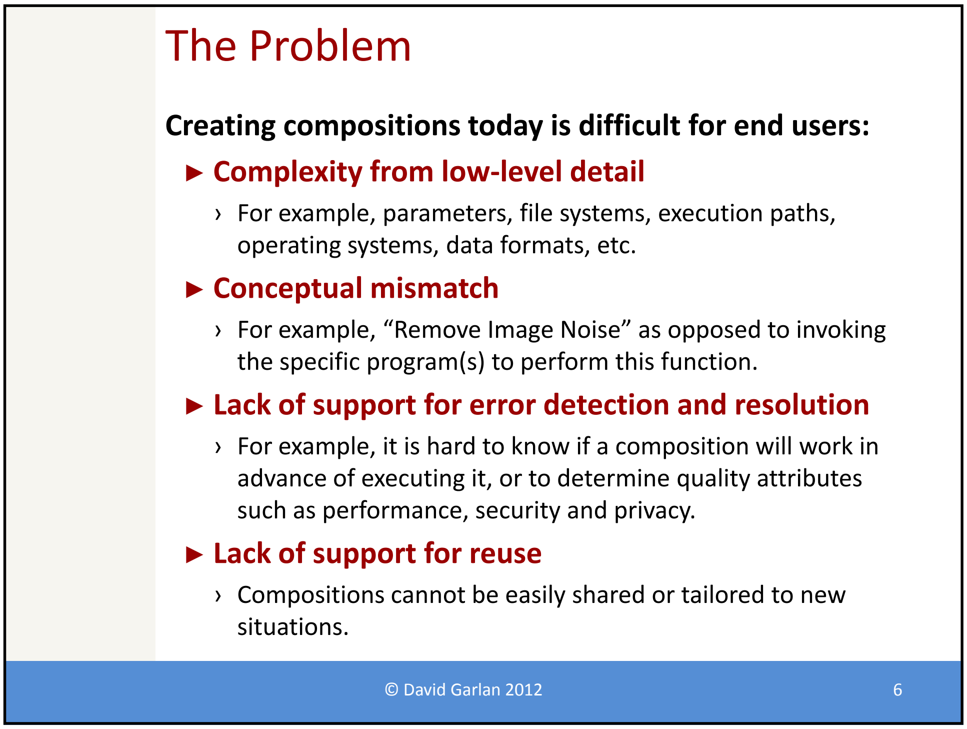


Business Process Management: Compose, analyze, reengineer and execute business processes.

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

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

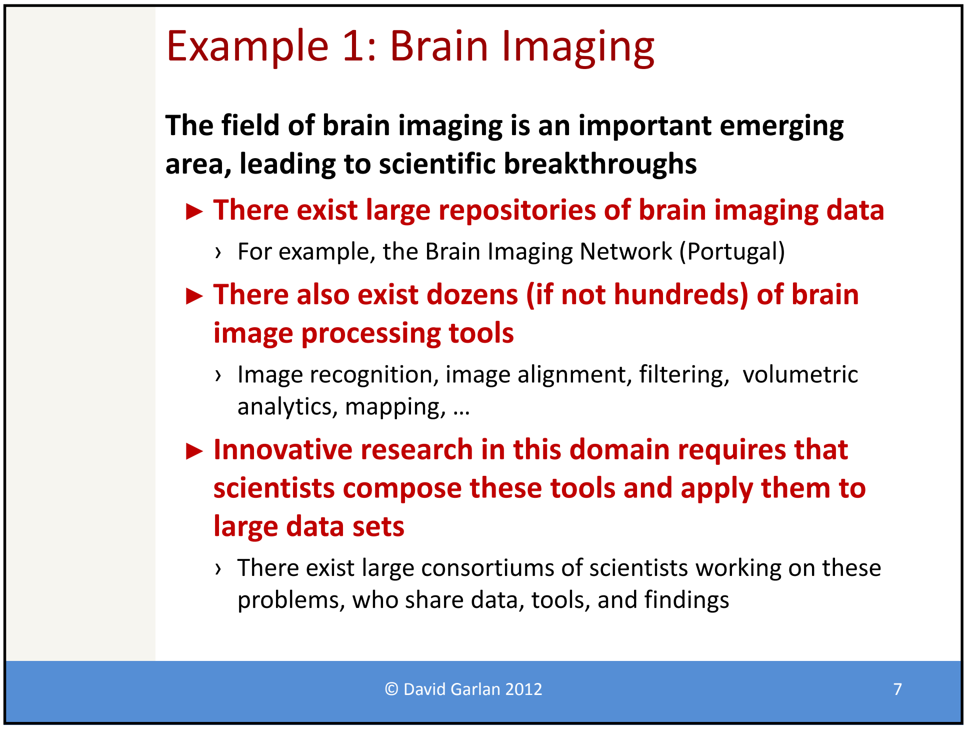
Digital Audio: Compose virtual audio components to synthesize music.



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.

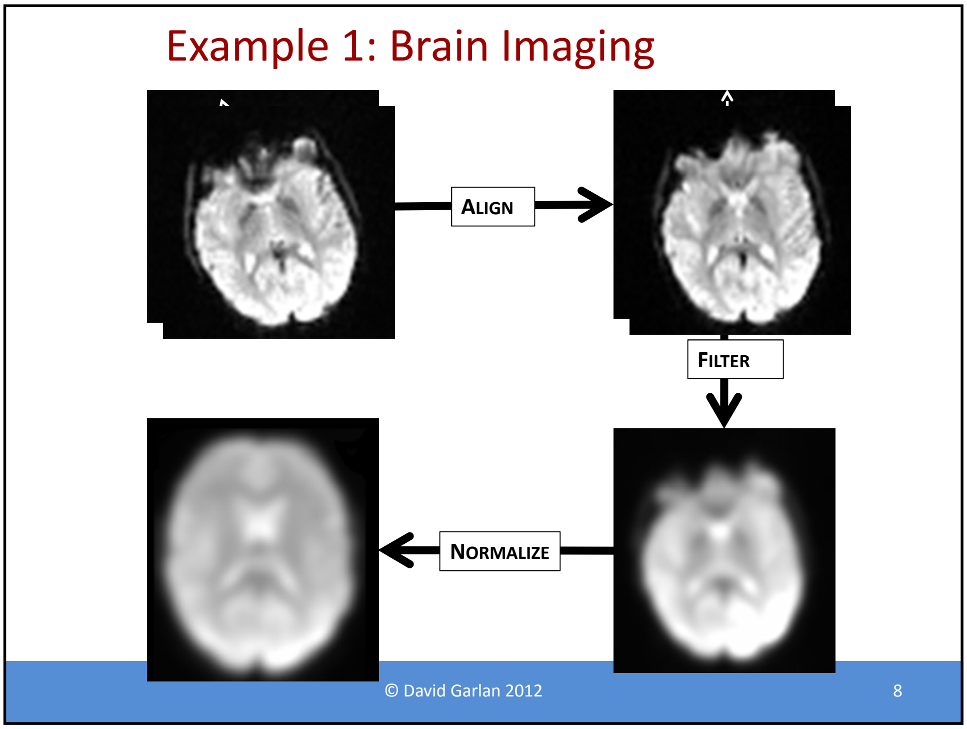


Example 1: Brain Imaging

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

© David Garlan 2012 7



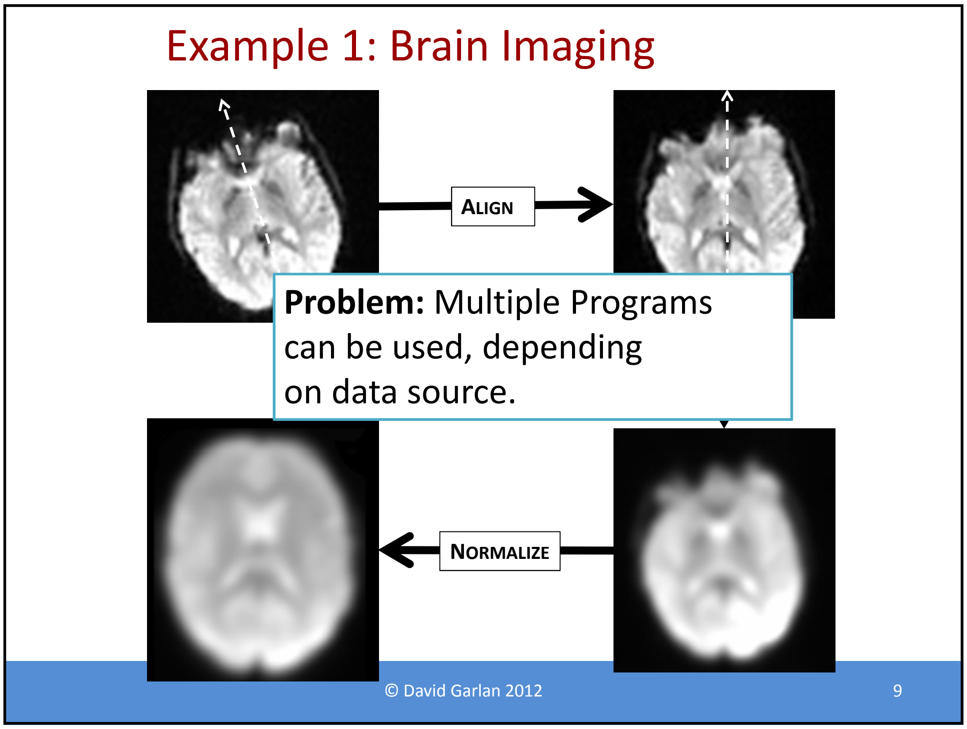
Example 1: Brain Imaging

ALIGN

FILTER

NORMALIZE

© David Garlan 2012 8

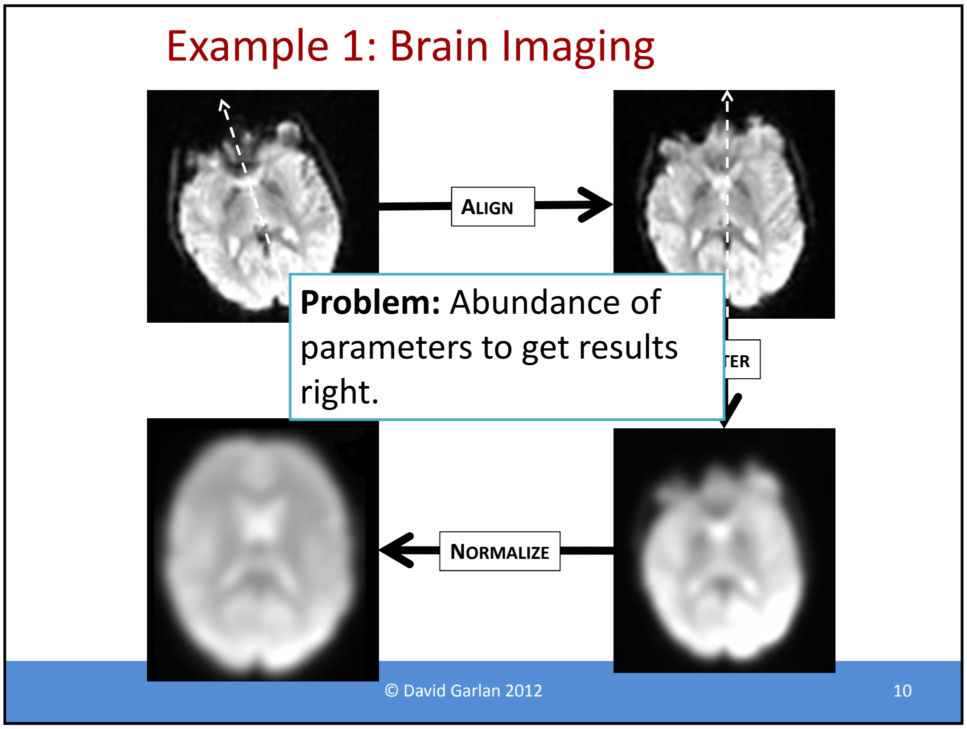


Example 1: Brain Imaging

Problem: Multiple Programs can be used, depending on data source.

© David Garlan 2012

9

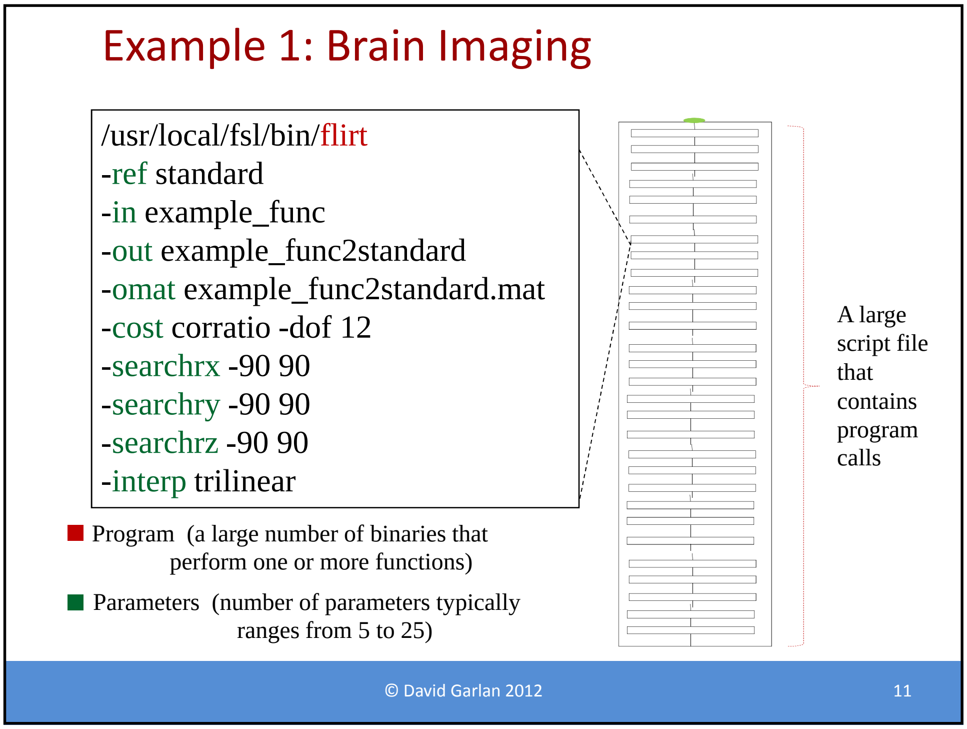


Example 1: Brain Imaging

Problem: Abundance of parameters to get results right.

© David Garlan 2012

10



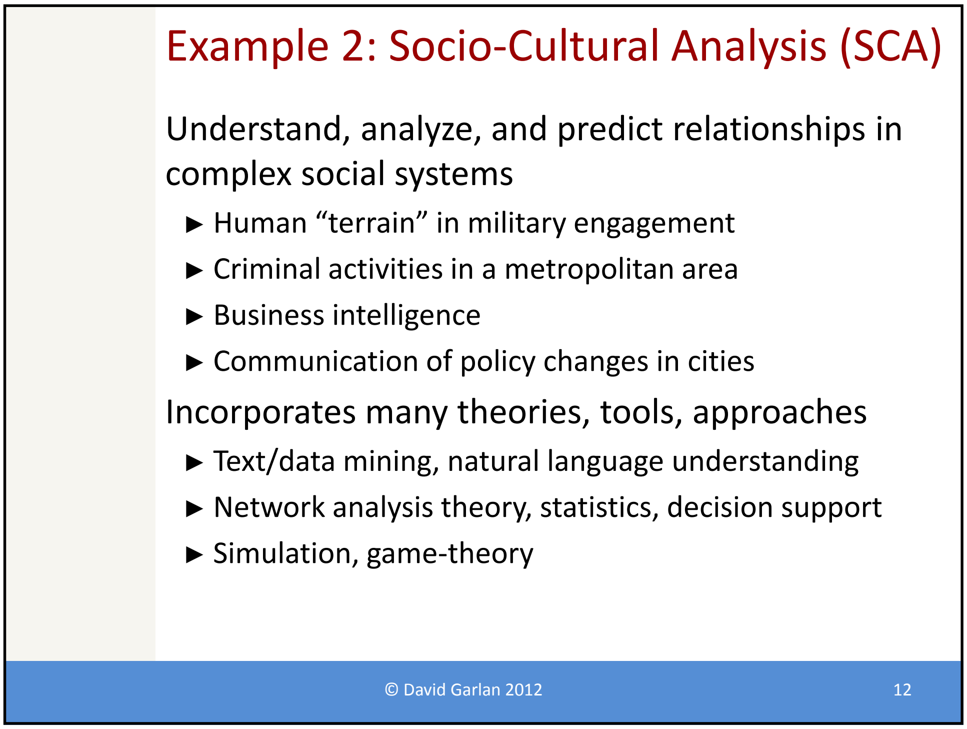
Example 1: Brain Imaging

```
/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
```

🟥 Program (a large number of binaries that perform one or more functions)

🟩 Parameters (number of parameters typically ranges from 5 to 25)

A large script file that contains program calls



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

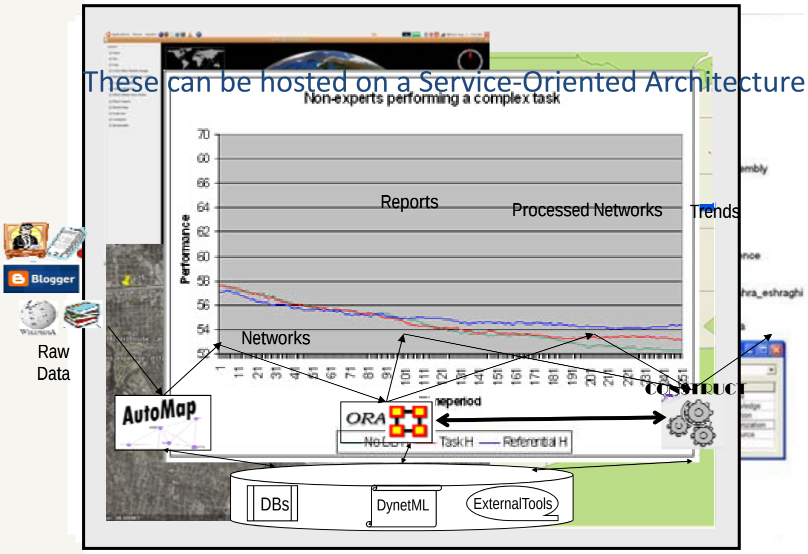
- Text/data mining, natural language understanding
- Network analysis theory, statistics, decision support
- Simulation, game-theory

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?

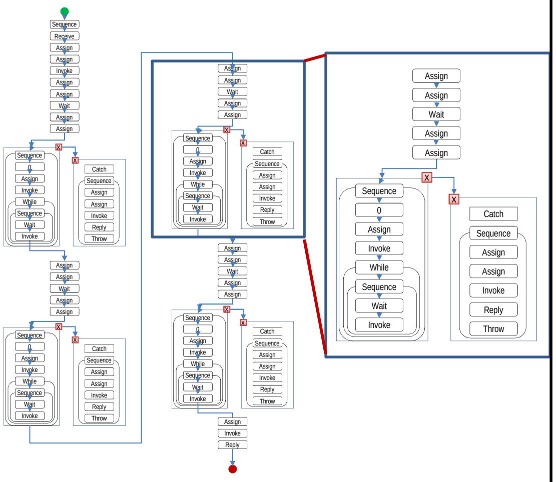
Using SCA Tools



These can be hosted on a Service-Oriented Architecture

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

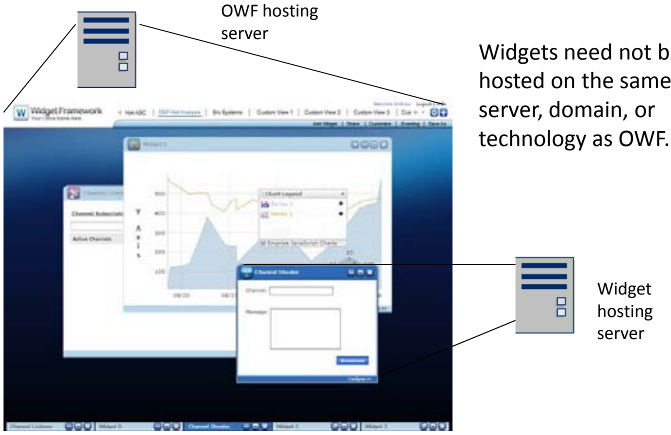
Example 3: Intelligence Analysis

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 for intelligence analysis – similar to 'Android Market'.

Ozone dashboard for displaying widgets

Deployment for Ozone Widgets

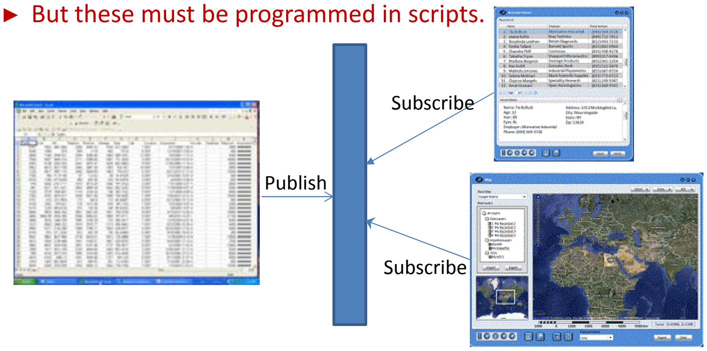


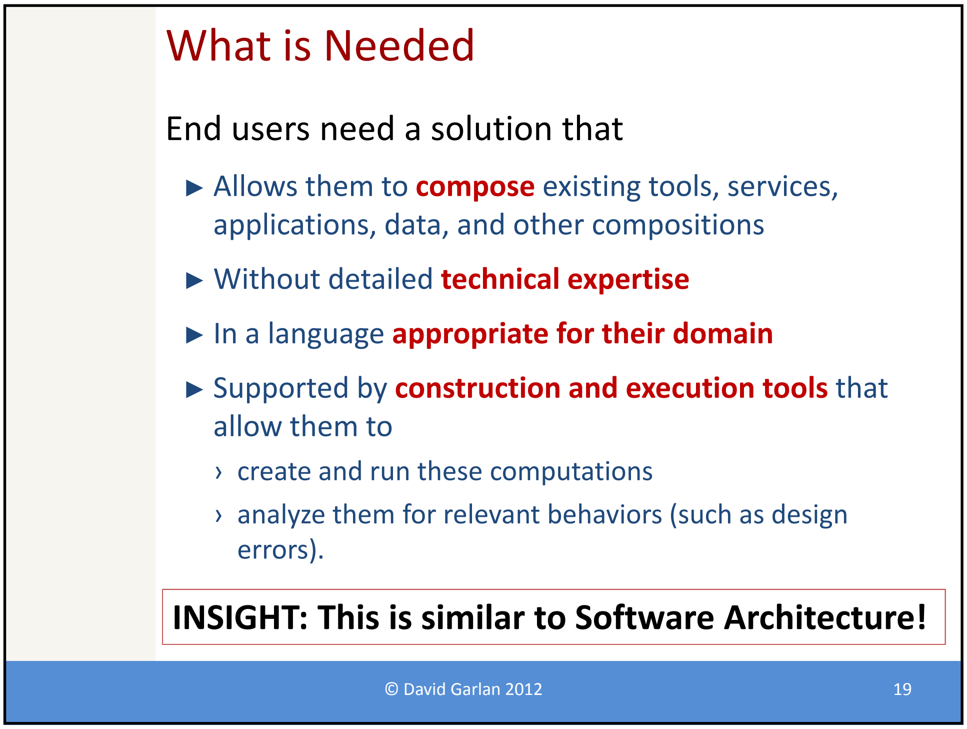
Widgets need not be hosted on the same server, domain, or technology as OWF.

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.





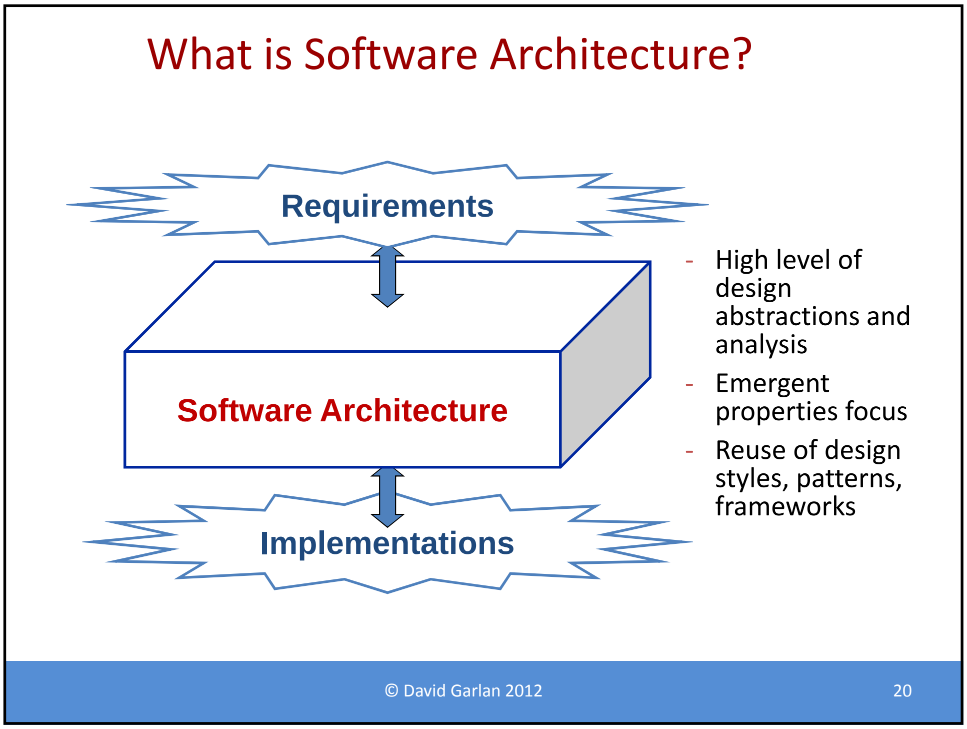
What is Needed

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!

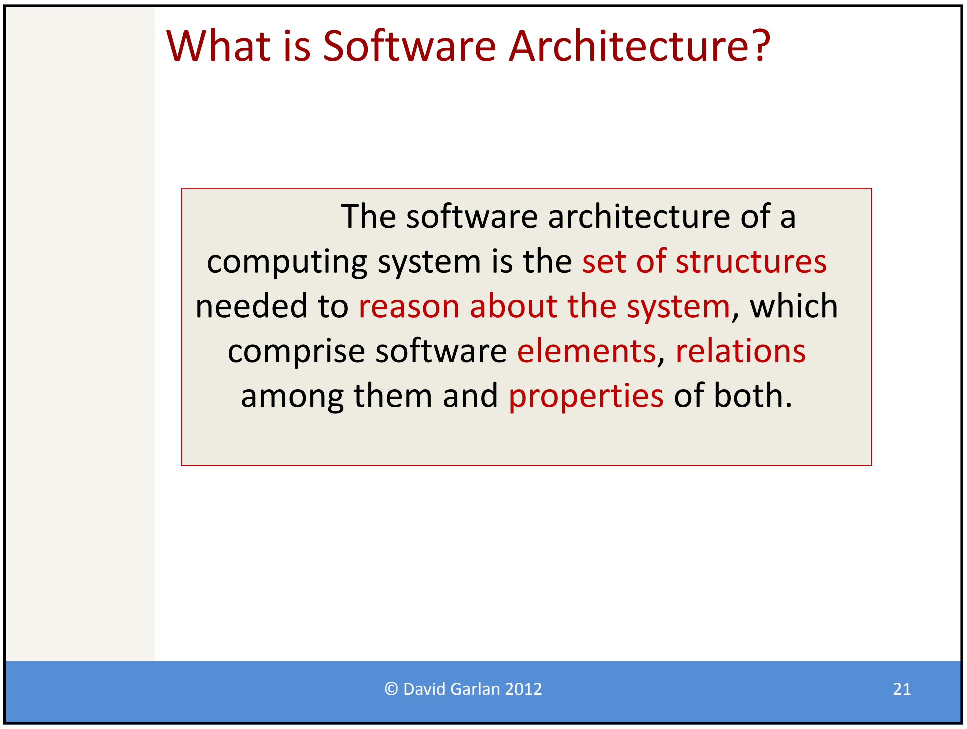
© David Garlan 2012 19



What is Software Architecture?

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

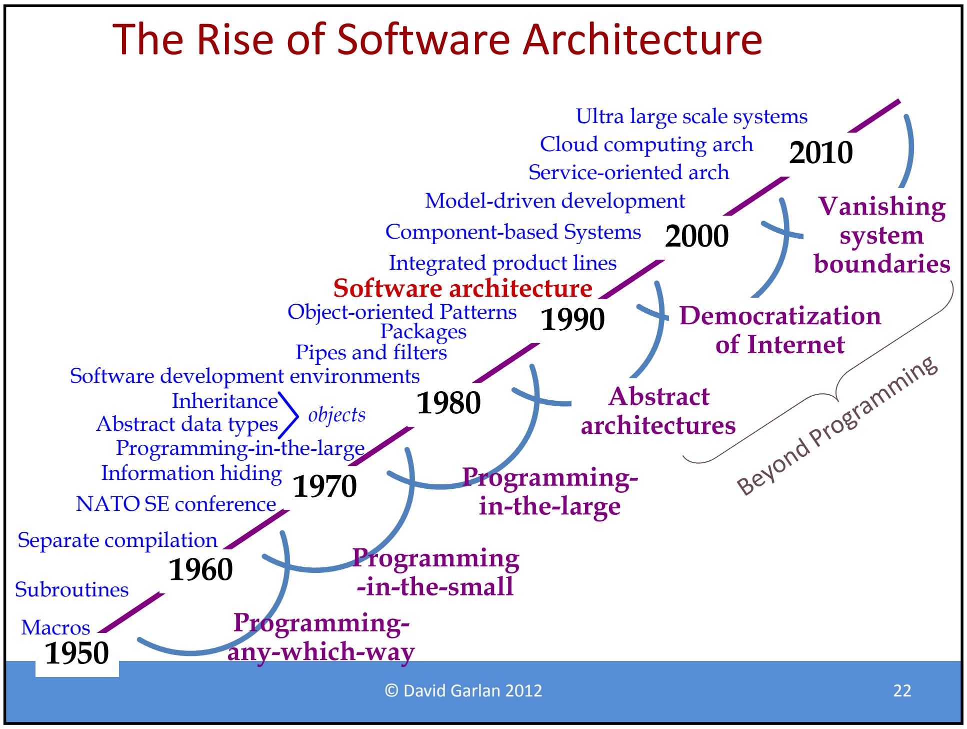
© David Garlan 2012 20



What is Software Architecture?

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.

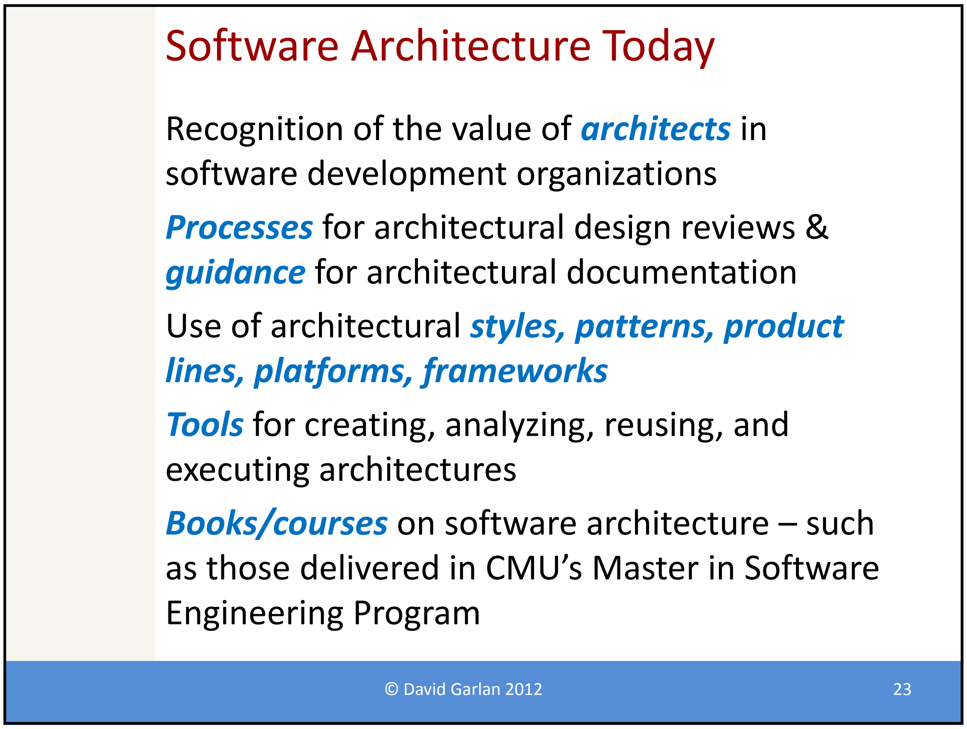
© David Garlan 2012 21



The Rise of Software Architecture

Ultra large scale systems
Cloud computing arch
Service-oriented arch
2010
Vanishing system boundaries
Model-driven development
Component-based Systems
2000
Integrated product lines
Software architecture
Object-oriented Patterns
Packages
Pipes and filters
1990
Democratization of Internet
Software development environments
Inheritance
Abstract data types
objects
1980
Abstract architectures
Beyond Programming
Programming-in-the-large
Information hiding
NATO SE conference
1970
Programming-in-the-large
Separate compilation
1960
Programming -in-the-small
Subroutines
Macros
1950
Programming-any-which-way

© David Garlan 2012 22



Software Architecture Today

Recognition of the value of ***architects*** in software development organizations

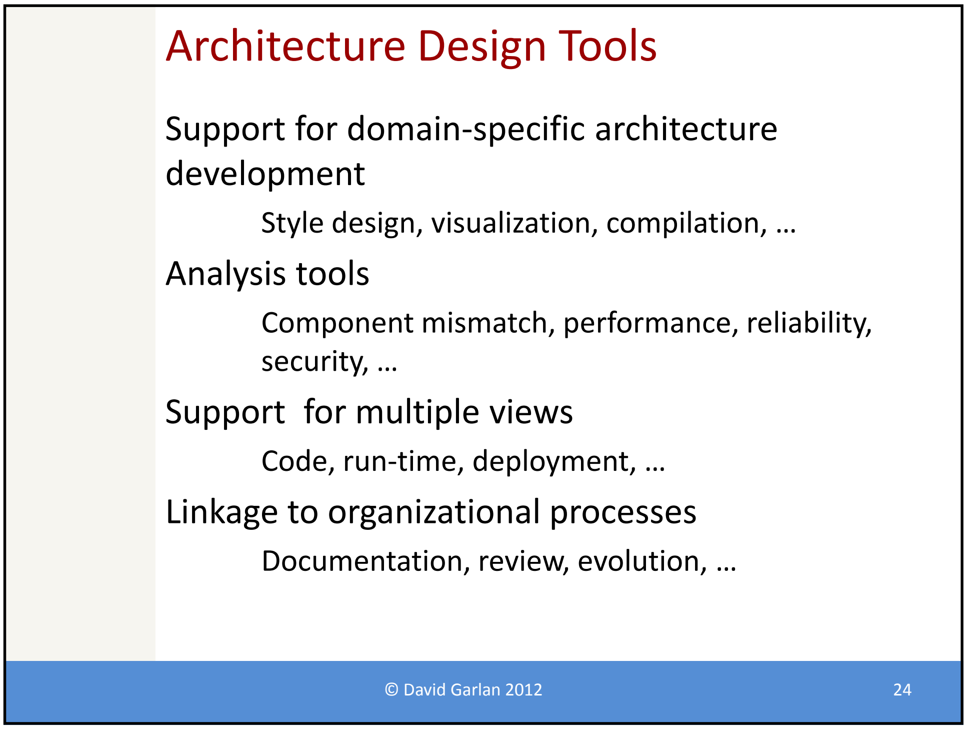
Processes for architectural design reviews & ***guidance*** for architectural documentation

Use of architectural ***styles, patterns, product lines, platforms, frameworks***

Tools for creating, analyzing, reusing, and executing architectures

Books/courses on software architecture – such as those delivered in CMU's Master in Software Engineering Program

© David Garlan 2012 23



Architecture Design Tools

Support for domain-specific architecture development

- Style design, visualization, compilation, …

Analysis tools

- Component mismatch, performance, reliability, security, …

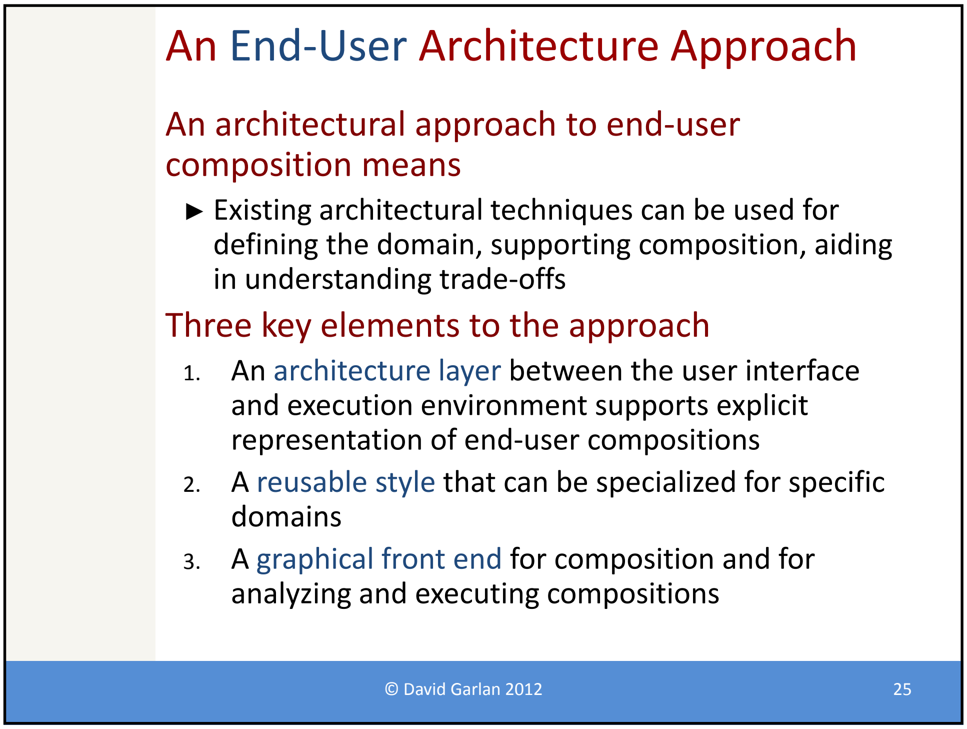
Support for multiple views

- Code, run-time, deployment, …

Linkage to organizational processes

- Documentation, review, evolution, …

© David Garlan 2012 24



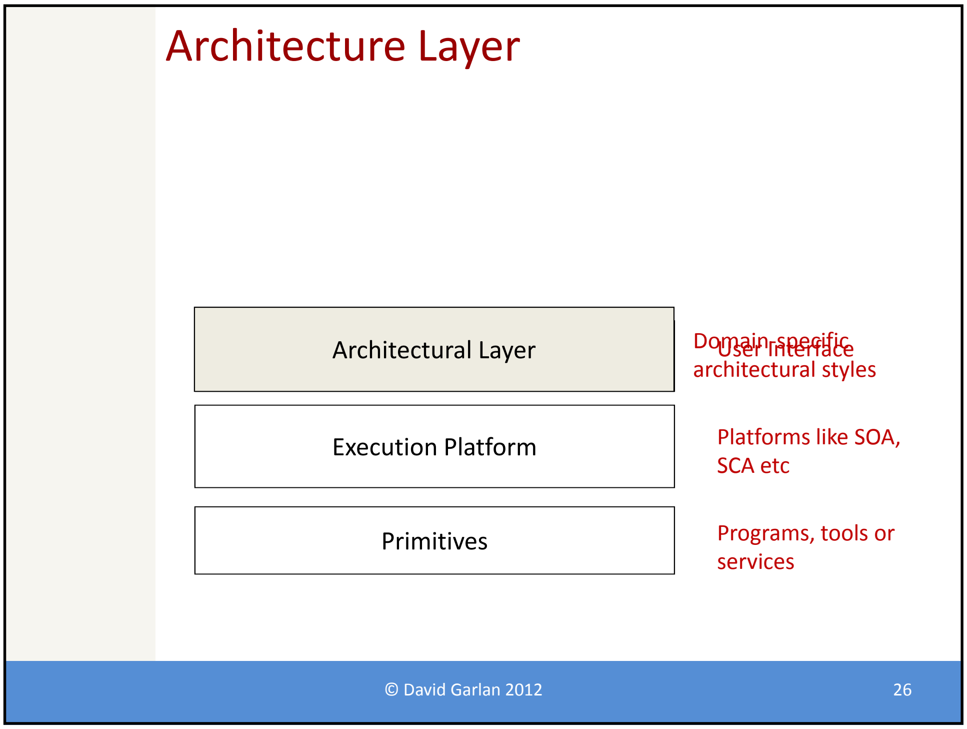
An End-User Architecture Approach

An architectural approach to end-user composition means

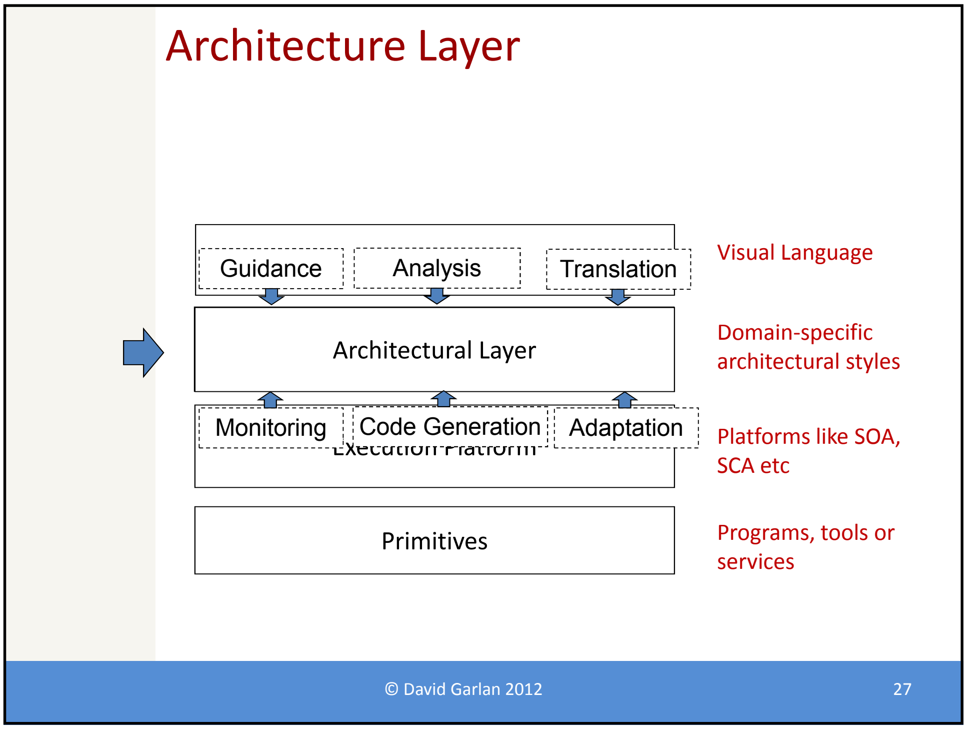
- ► Existing architectural techniques can be used for defining the domain, supporting composition, aiding in understanding trade-offs

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



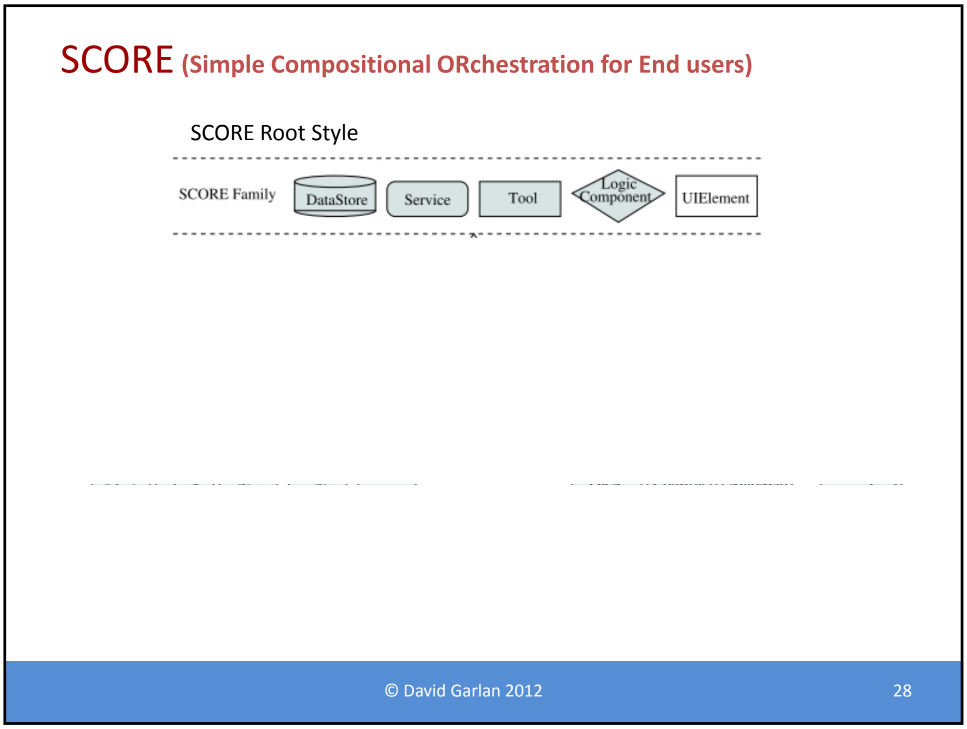
Architecture Layer

Visual Language

Domain-specific architectural styles

Platforms like SOA, SCA etc

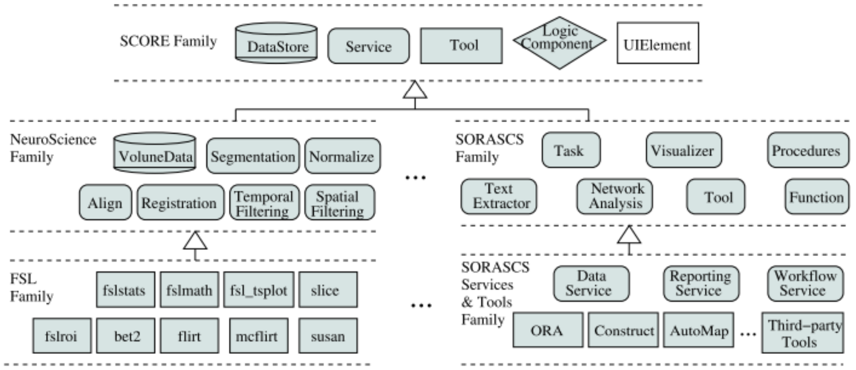
Programs, tools or services



SCORE (Simple Compositional ORchestration for End users)

SCORE (Simple Compositional ORchestration for End users)

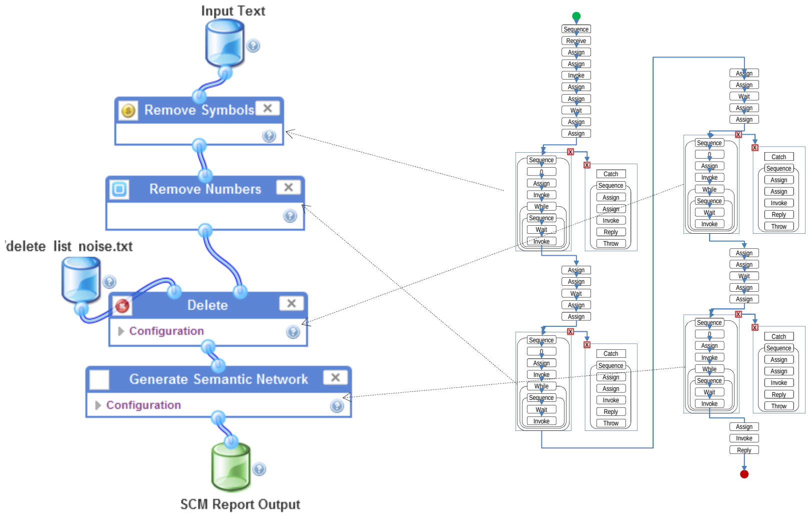
SCORE Root Style



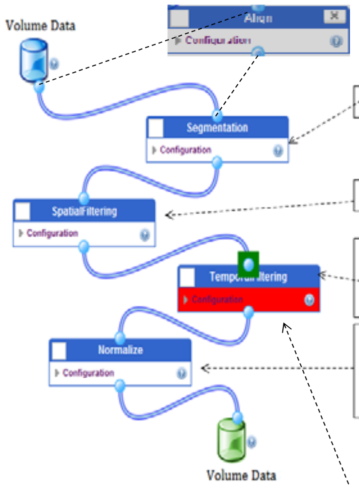
SCORE Specialization for Neuroscience

SCORE Specialization for Intelligence Analysis

Specializing SCORE: Intelligence Analysis



Specializing SCORE: Neuroscience



Service Implementation

```
bet2 ${2} ${4} -f ${3} -n -m
```

```
fslmath ${2} -kernel gauss ${sigma} -fmean ${5}
```

```
hp=`echo "scale=10;100/${3}" | bc`
lp=`echo "scale=10;3/${3}" | bc`
fslmath ${2} -bptf ${lp} -1 -mas mask ${6}
```

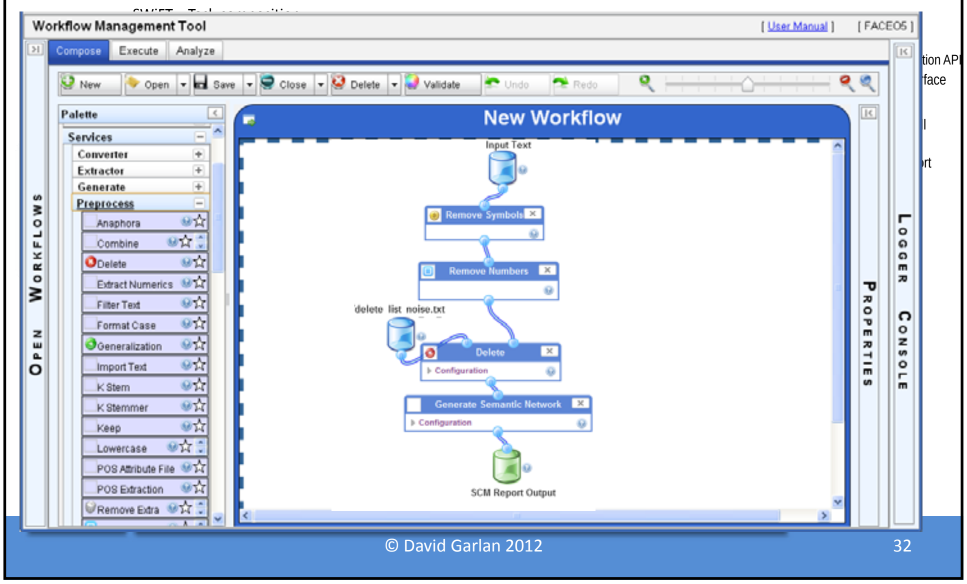
```
flirt -ref standard -in ${2} -out
${input_in_standard} -omat ${input2standard}.mat
-cost corratio -dof 12 -searchrx -90 90 -
searchry -90 90 -searchrz -90 90 -interp
trilinear
```

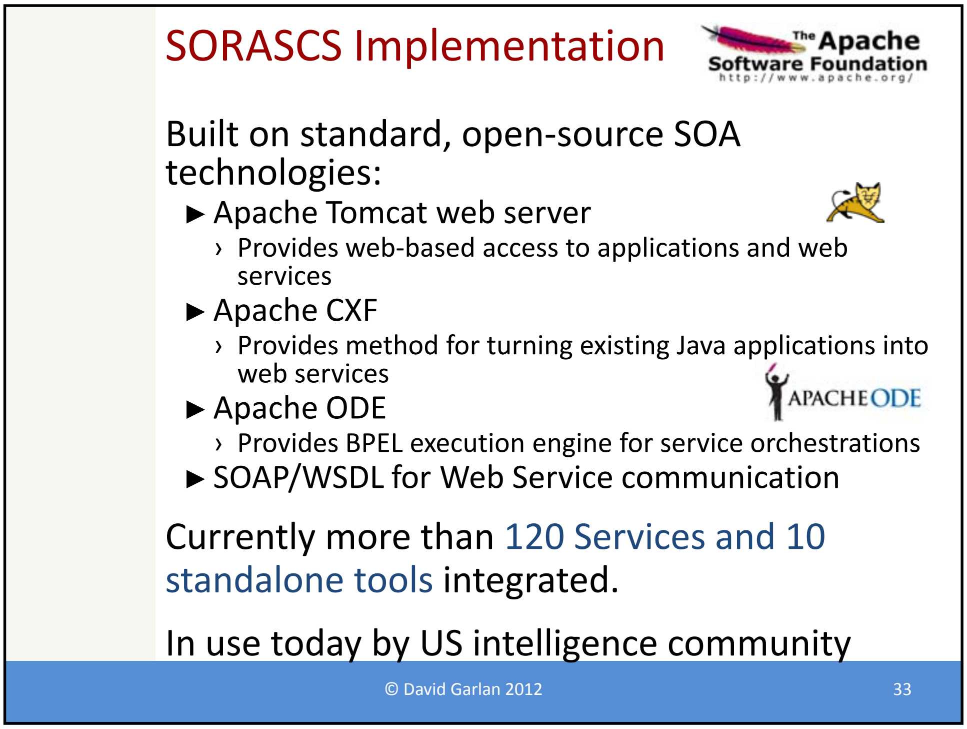
FSL Components

- fslroi
- mcflirt
- bet2
- fslmath
- flirt
- …

“Missing Alignment before Temporal Filtering”

SORASCS System Organization





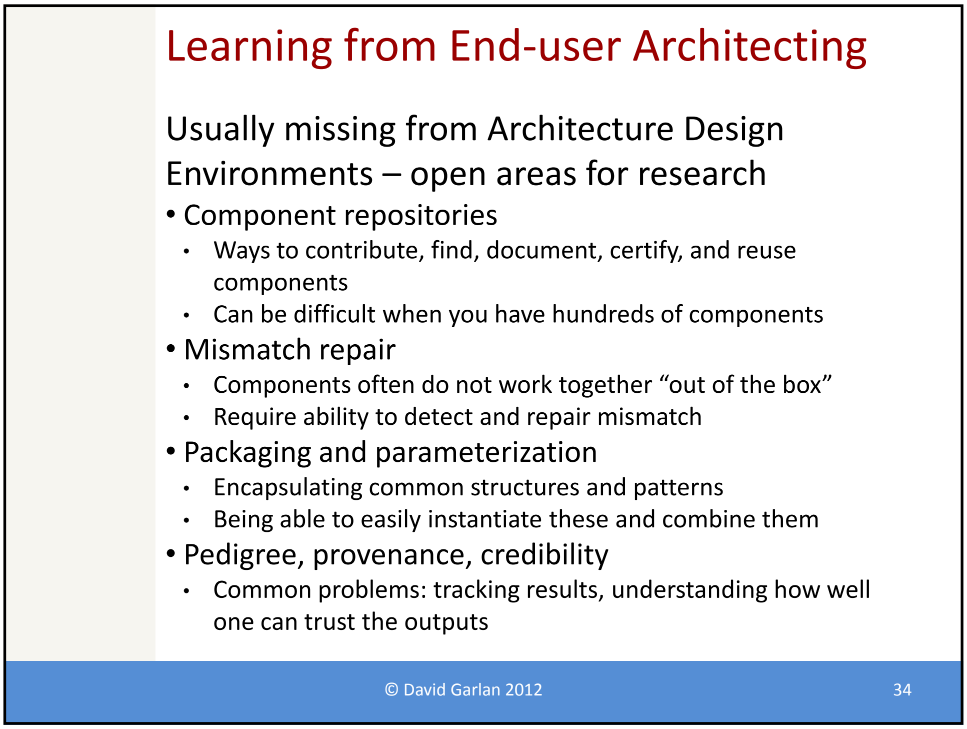
SORASCS Implementation

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

Beyond Architecture: Sustainability

It is not enough to have a good platform, interface, and set of components

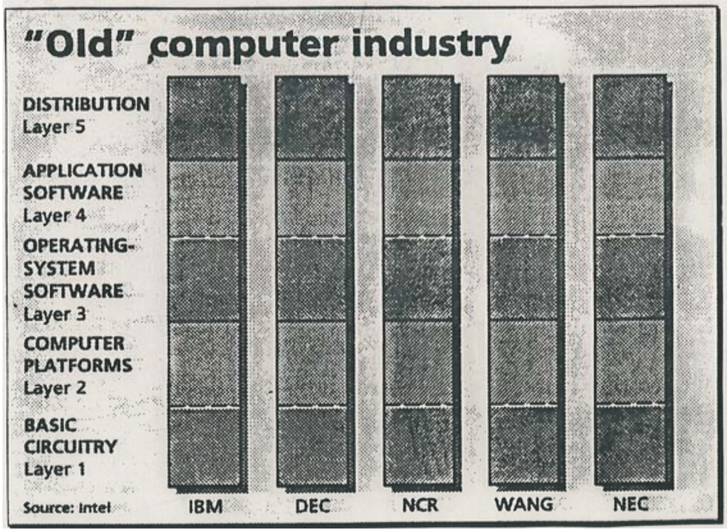
To be successful we require **sustainability**

This, in turn, requires a stable **Socio-Technical Ecosystem**

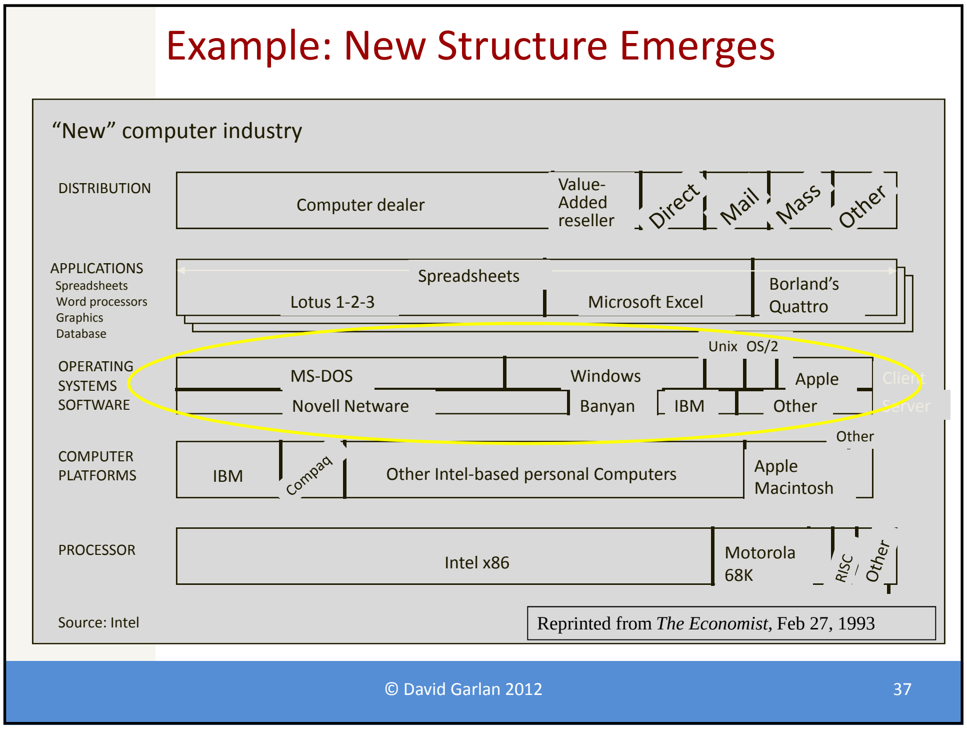
A **Socio-Technical Ecosystem (STE)** represents a complex, self-sustaining system including:

- **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

Example: Structure of the Mainframe Computer Industry



Reprinted from *The Economist*, Feb 27, 1993



Example: New Structure Emerges

"New" computer industry

DISTRIBUTION
Computer dealer
Value-Added reseller
Direct
Mail
Mass
Other

APPLICATIONS
Spreadsheets
Word processors
Graphics
Database

Spreadsheets
Lotus 1-2-3
Microsoft Excel
Borland's Quattro

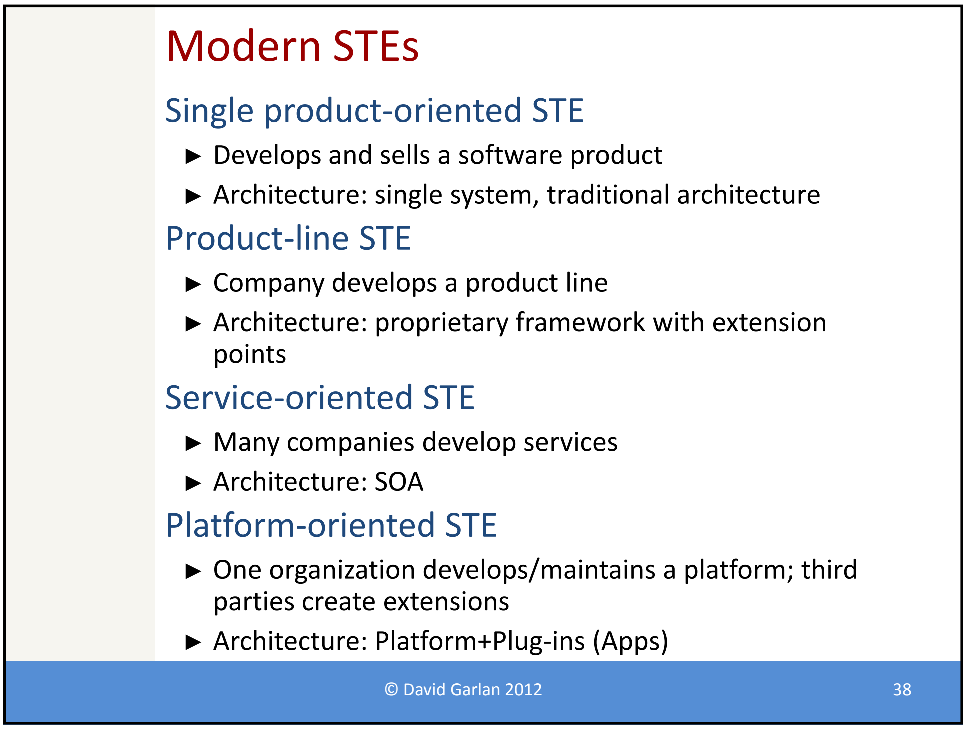
OPERATING SYSTEMS SOFTWARE
Unix OS/2
MS-DOS
Windows
Apple
Client
Novell Netware
Banyan
IBM
Other
Server

COMPUTER PLATFORMS
Other
IBM
Compaq
Other Intel-based personal Computers
Apple Macintosh

PROCESSOR
Intel x86
Motorola 68K
RISC
Other

Source: Intel

Reprinted from *The Economist*, Feb 27, 1993



Modern STEs

Single product-oriented STE

- ▶ Develops and sells a software product
- ▶ Architecture: single system, traditional architecture

Product-line STE

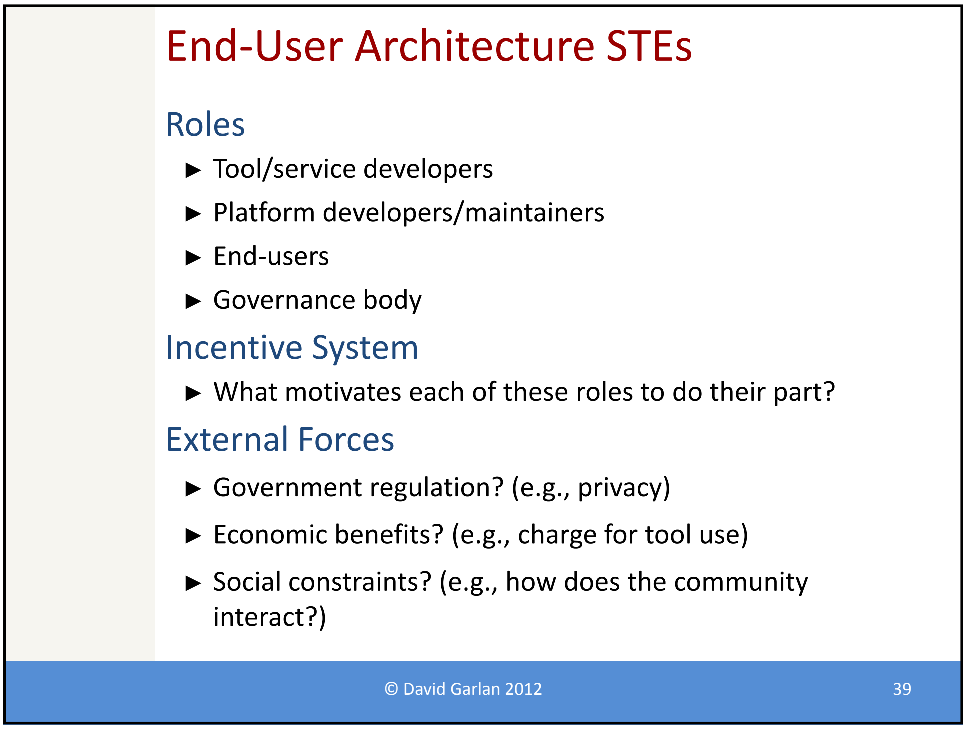
- ▶ Company develops a product line
- ▶ Architecture: proprietary framework with extension points

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

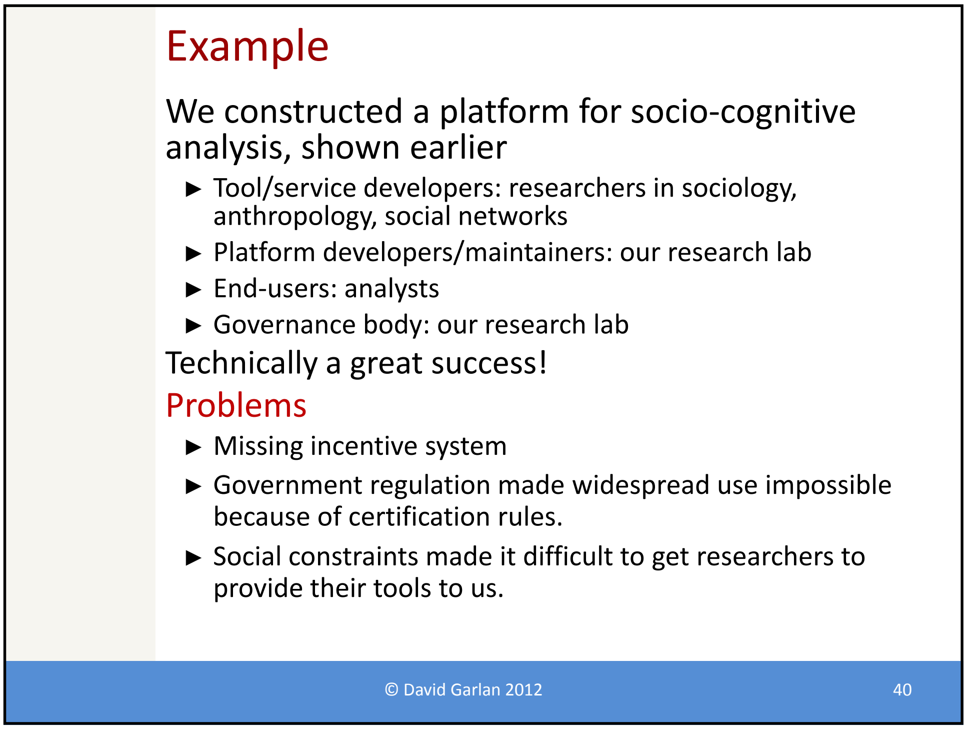
Incentive System

- 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?)

© David Garlan 2012 39



Example

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

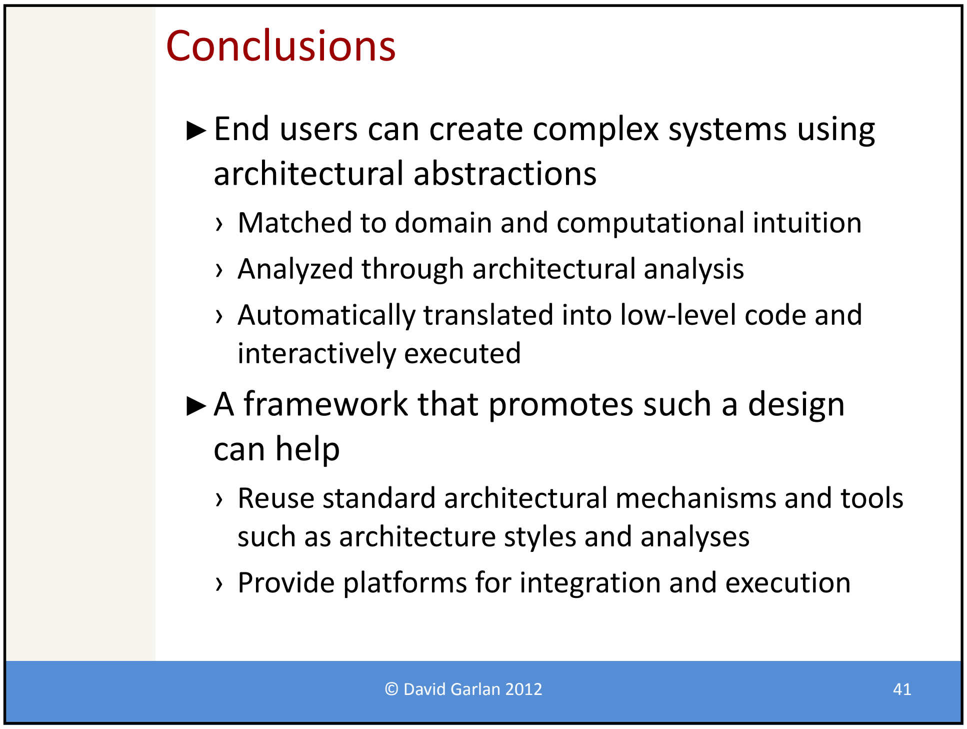
- Tool/service developers: researchers in sociology, anthropology, social networks
- Platform developers/maintainers: our research lab
- End-users: analysts
- Governance body: our research lab

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.

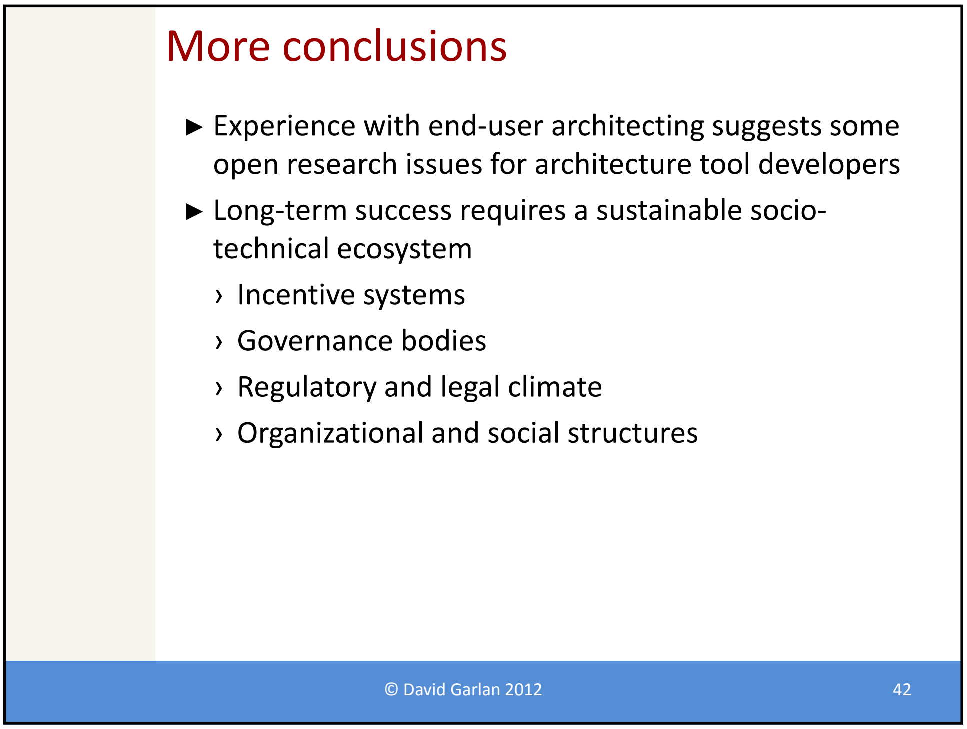
© David Garlan 2012 40



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

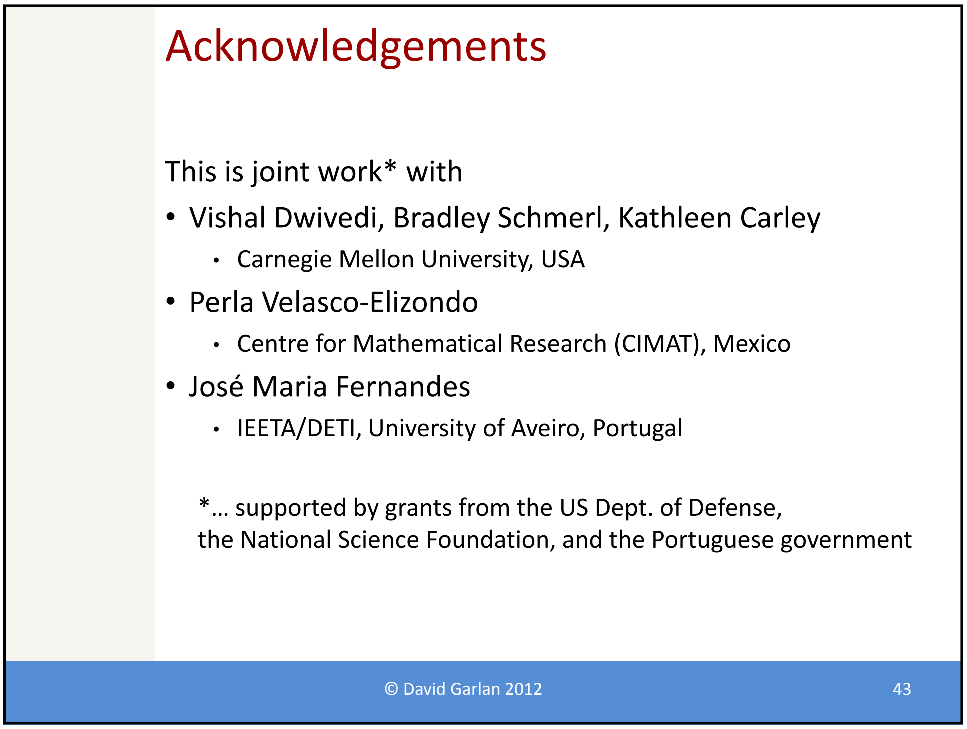
© David Garlan 2012 41



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

© David Garlan 2012 42

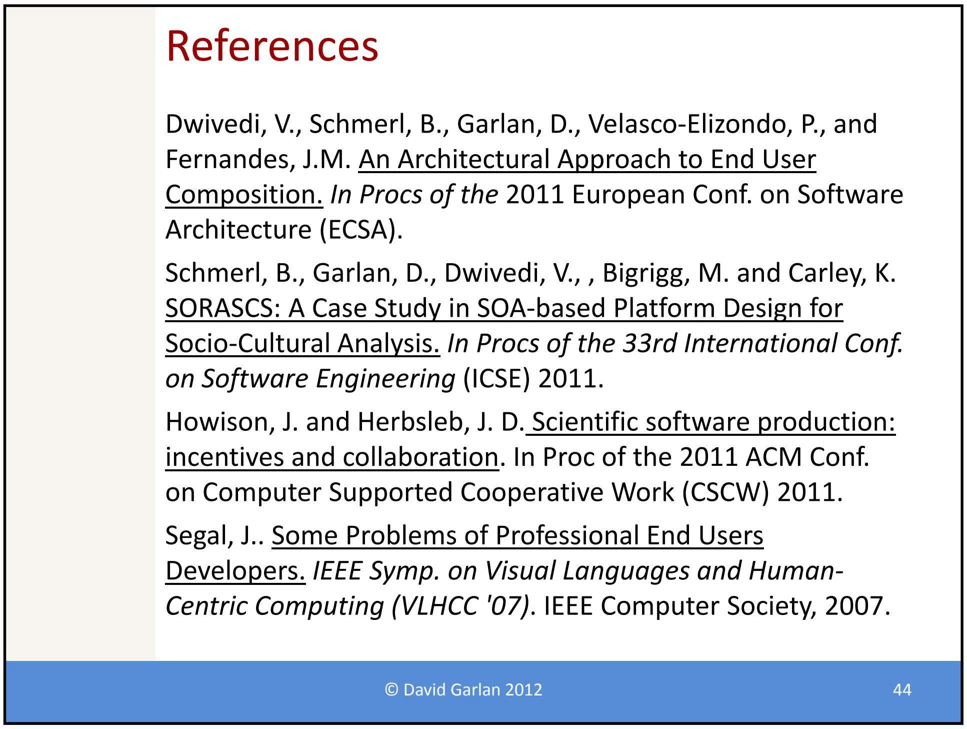


Acknowledgements

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



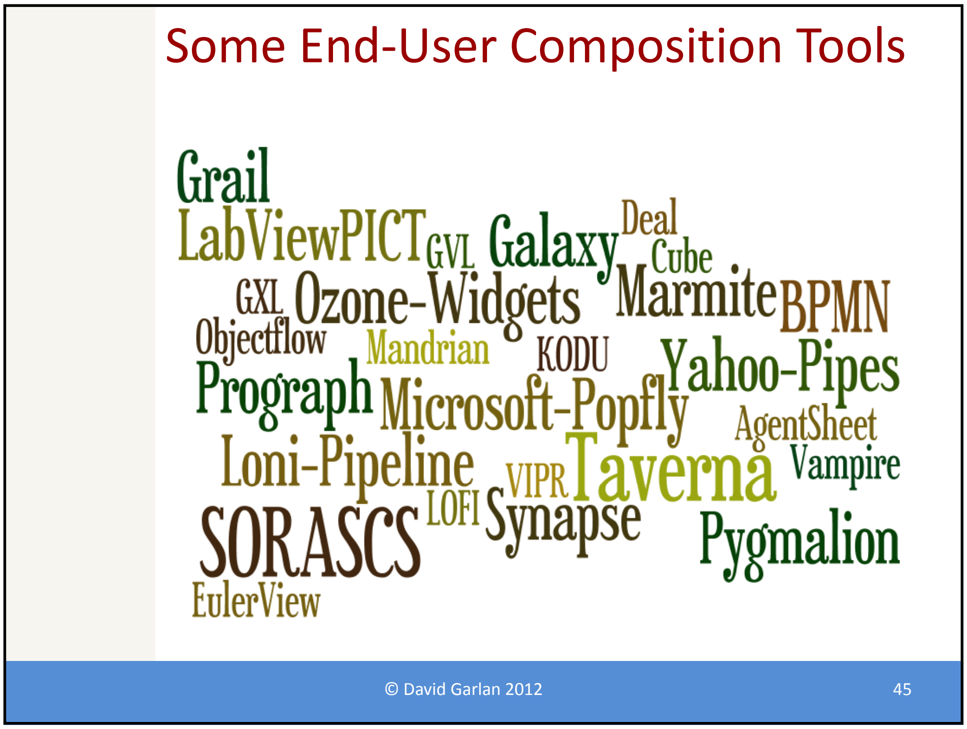
References

Dwivedi, V., Schmerl, B., Garlan, D., Velasco-Elizondo, P., and Fernandes, J.M. An Architectural Approach to End User Composition. *In Procs of the* 2011 European Conf. on Software Architecture (ECSA).

Schmerl, B., Garlan, D., Dwivedi, V., , Bigrigg, M. and Carley, K. SORASCS: A Case Study in SOA-based Platform Design for Socio-Cultural Analysis. *In Procs of the 33rd International Conf. on Software Engineering* (ICSE) 2011.

Howison, J. and Herbsleb, J. D. Scientific software production: incentives and collaboration. In Proc of the 2011 ACM Conf. on Computer Supported Cooperative Work (CSCW) 2011.

Segal, J.. Some Problems of Professional End Users Developers. *IEEE Symp. on Visual Languages and Human-Centric Computing (VLHCC '07)*. IEEE Computer Society, 2007.



Some End-User Composition Tools

Grail
LabView
PICT
GVL
Galaxy
Deal
Cube
GXL
Ozone-Widgets
Marmite
BPMN
Objectflow
Mandrian
KODU
Yahoo-Pipes
Prograph
Microsoft-Popfly
AgentSheet
Loni-Pipeline
VIPR
Taverna
Vampire
SORASCS
LOFI
Synapse
Pygmalion
EulerView

© David Garlan 2012 45