



Architecture-related Research:
Architecture Reconstruction and Scope, Cost and Effort Estimation

Liam O'Brien
 ANU Lecture

NICTA

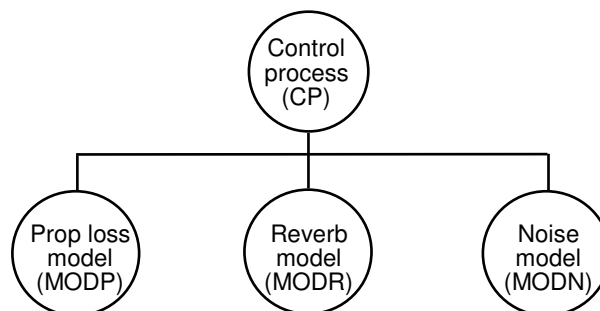
Australian Government
 Department of Broadband, Communications and the Digital Economy
 Australian Research Council

NICTA Members
 ANU
 UNSW
 Department of State and Regional Development
 Victoria
 The University of Sydney
 Queensland Government
 Griffith University
 Curtin University
 The University of Queensland

NICTA Partners

2

- Architecture Reconstruction
- SMAT-AUS – Scope, Cost and Effort Estimation Framework



What's wrong with the Diagram?



- Many things left unspecified:
 - What kind of components?
 - What kind of connectors?
 - What do the circles and lines mean?
 - What is the significance of the layout?
 - Why is control process (CP) on a higher level?
- Circle and line (box and arrow) drawings alone are not architectures; rather, they are a starting point.

Definition of Software Architecture



The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

- *Software Architecture in Practice*
Bass, Clements, Kazman

Implications of the Definition



- Architecture is an abstraction of systems.
 - architecture defines components and how they interact.
 - architecture suppresses purely local information about components; private details are not architectural.
- Systems have many views.
 - no single view can be *the* architecture.
 - the set of candidate views is not fixed or prescribed: whatever is useful for analysis, communication, or understanding.

Architectural Views 1



- In a house, there are plans for
 - rooms
 - electrical wiring
 - plumbing
 - ventilation
- Each of these constitutes a “view” of the house.
 - used by different people
 - used to achieve different qualities in the house
 - serves as a description and prescription
- So it is with software architecture.

- Which views are used, and why?
- Some views from the literature:

Class	Concurrency	Implementation
Physical	Development	Logical
Uses	Module	Collaboration
Calls	Functional	
Data flow	Deployment	

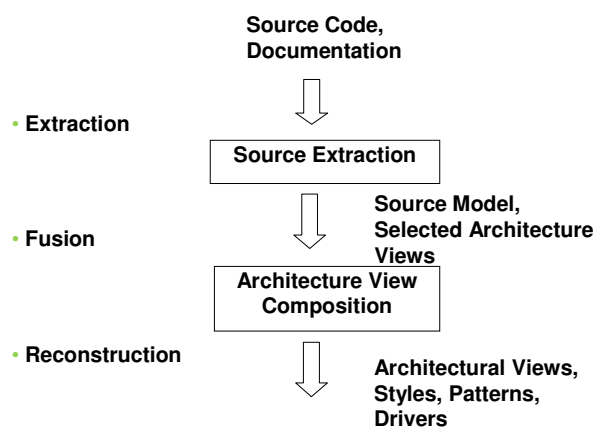
- Architectures are frequently undocumented or documentation is not current.
- Architectural drift and erosion are unavoidable.
- However, we need to be able to reason about the architectures of *existing* systems:
 - for conformance evaluation
 - for reuse
 - in support of product line development
 - for analysis

So What Is Architecture Reconstruction?



- A process in which cultural hypotheses are generated and tested
- These hypotheses are ideally the inverse of the mappings developed during design
- The reconstructor must add information during the reconstruction process
 - This depends on the reconstructor's available information and bias

Reconstruction Process



- Apply whatever tools are available/appropriate/necessary for a given target system:
 - parsers (Imagix, SNIFF+, CIA, rigiparse, Understand)
 - AST-based analyzers (Gen++, Refine)
 - lexical analyzers (LSME-lightweight source model extraction)
 - profilers (gprof)
 - code instrumentation
 - ad hoc (grep, perl)

- Much architecture-related information may be extracted *statically* from source code, compile-time artifacts and design artifacts.
- Some architecturally relevant information may not, due to *late binding*:
 - polymorphism
 - function pointers
 - run-time parameterization

Extraction: Source Model



- Source model consists of:
 - Collection of source elements
 - e.g. files, functions, variables, classes, methods
 - Set of relations among elements
 - function **calls** function, function **accesses** variable
 - Attributes
 - function calls function N times

Typical Elements and Relations



- Some elements and relations relevant to software architecture:
 - function **calls** function
 - function **accesses** variable
 - file **contains** function
 - class **has_subclass** class
 - class **has_friend** class
 - process **communicates_with** process
 - process **writes** file
 - ...

- Architectural elements are not explicitly represented in source code
 - “user interface”, “repository”, “cache”, etc. are usually tangled collections of functions or objects
 - because of the sequence of mappings applied, architectural constructs are realized by *many* mechanisms in an implementation
- Therefore, we need to “reconstruct” architectural elements by applying the inverses of the mappings

- Architecture reconstruction carried out through:
 - Aggregation
 - Pattern matching
 - Analyzing documents (including code)
 - Interviewing maintainers/developers
 - ...

- **Interpretive, iterative** and **interactive**.
- *Not* automatic.
- How?
 - Query-based pattern matching for:
 - typing
 - clustering
- Direct manipulation

- Primary mechanism for manipulation is the application of patterns (i.e. inverse mappings).
- Examples:
 - aggregate local variables with functions
 - aggregate members with classes
 - compose architecture-level elements

Case Study Example



- The system in the case study is an embedded automotive control system.

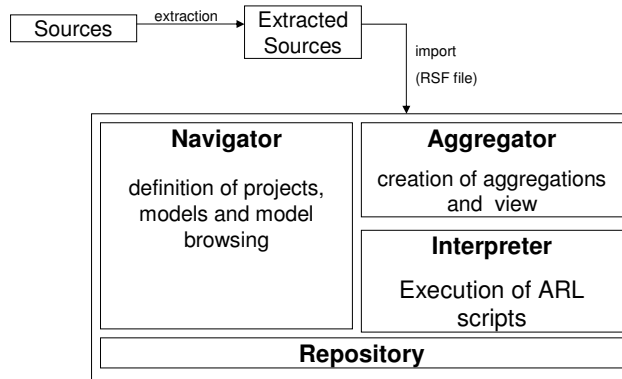
	Files	KLOC	Functions	Macros	Variables	Types
C/c	61	9	376	225	180	5
Header	71	2	0	1055	311	56
Total	132	11	376	1280	491	61

Source Model Extracted



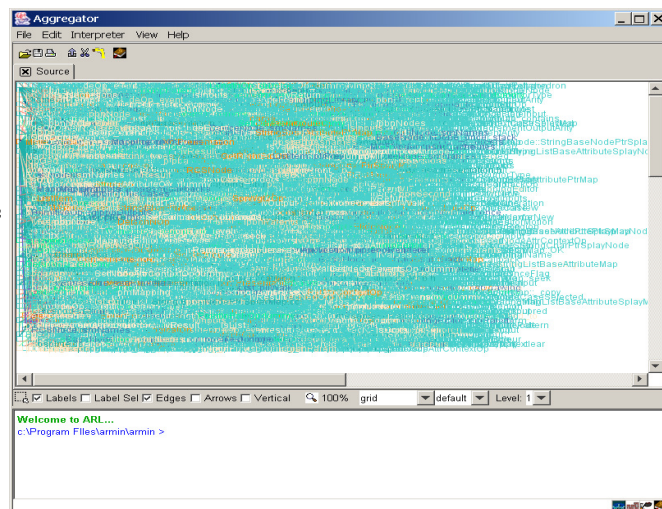
Source	Relation Type	Target	Description
File	includes	File	A C #include of one file by another
File	contains	Function	A definition of a function in a file
File	defines_var	Variable	A definition of a variable in a file
File	defines_macro	Macro	A definition of a macro in a file
Function	calls	Function	A static function call
Function	accesses_var	Variable	A function access on a variable
Function	accesses_macro	Macro	A function access on a macro
Component	in_comp	File	A file belongs to a component
Layer	in_layer	Component	A component belongs to a layer

Architecture Reconstruction and MINING (ARMIN) Tool



View of all Elements and Relations

- “White noise”:
- A typical picture of raw extracted views.



- **File aggregation:**

```
$d = list(system.types.contains);  
$d.merge(/ext="+");  
$d1 = collapse($d,/graph="FILE+",/type=system.types.file);  
$d.remove();  
show();
```

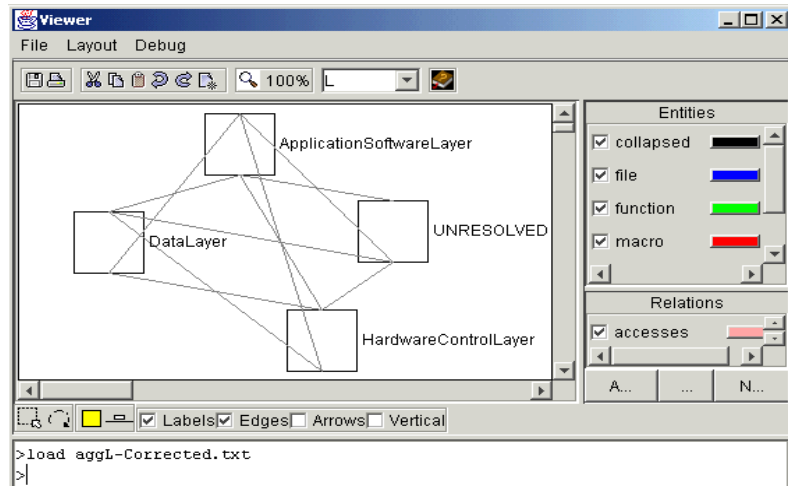
- **Module/Component aggregation:**

```
$e = list(system.types.in_comp);  
$e.merge(/ext="+");  
$e1 = $d1.collapse($e,/graph="COMP+",  
/type=system.types.component);  
$e.remove();  
show();
```

- **Layer aggregation:**

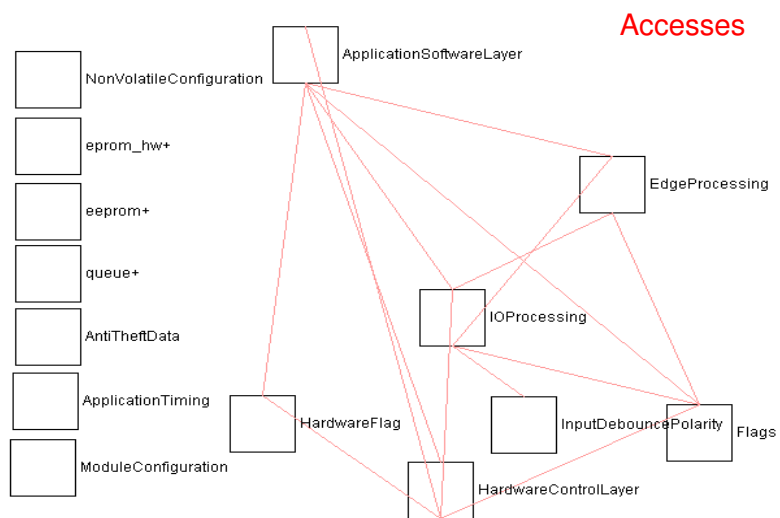
```
$f = list(system.types.in_layer);  
$f.merge(/ext="+");  
$f1 = $e1.collapse($f,/graph="LAYER+",  
/type=system.types.layer);  
$f.remove();  
show();
```

High-level Layer view



ANU Lecture – Architecture-related Research

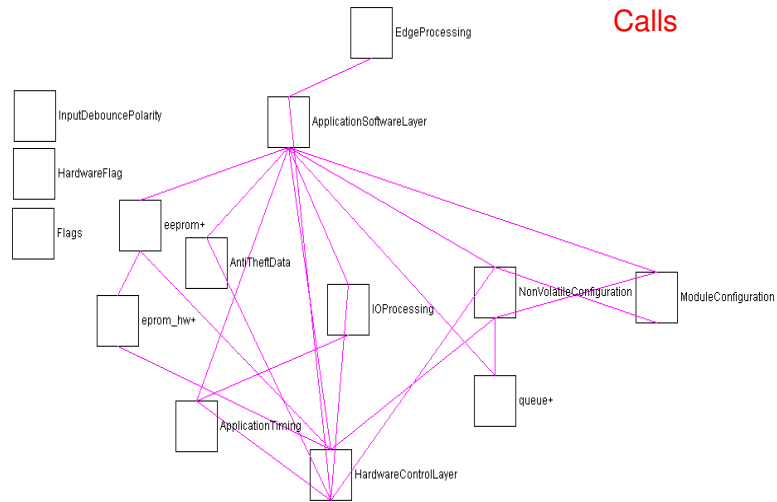
Exploring the Data Layer 1



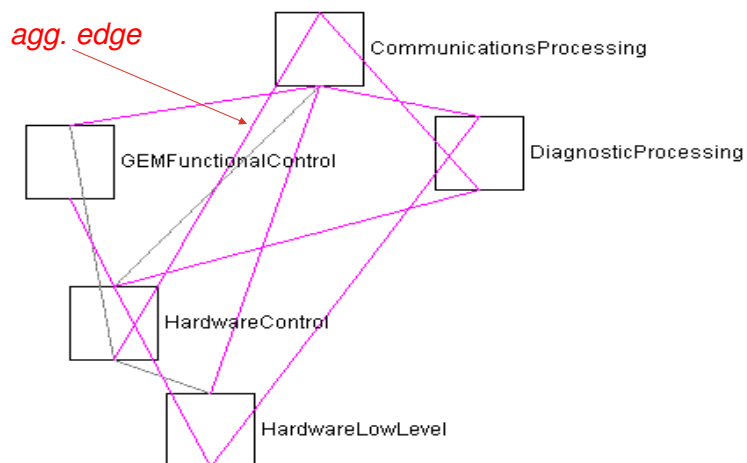
ANU Lecture – Architecture-related Research

28

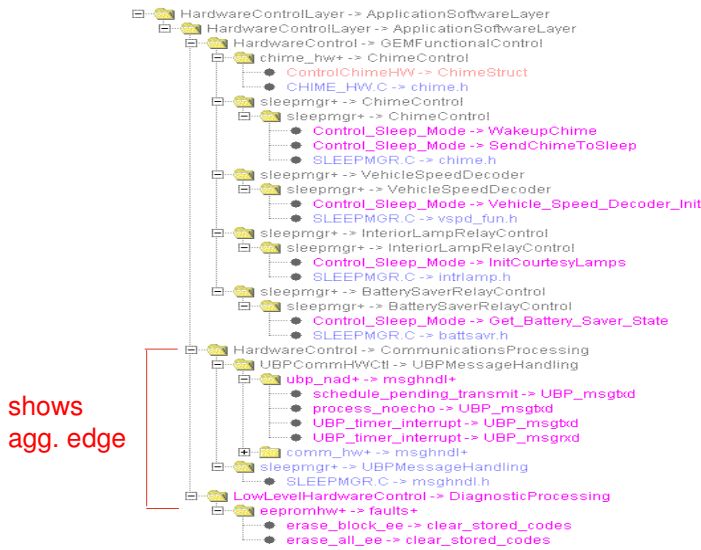
Exploring the Data Layer 2



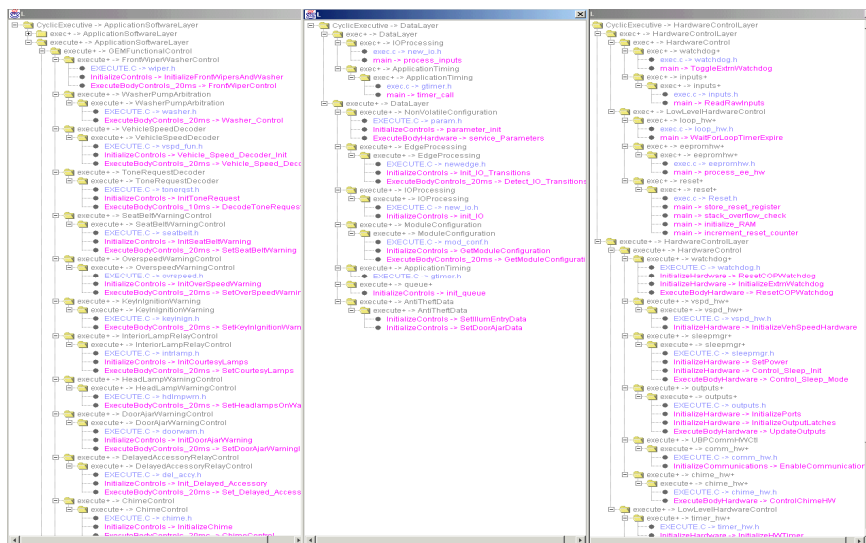
Exploring the views (SW and HW)



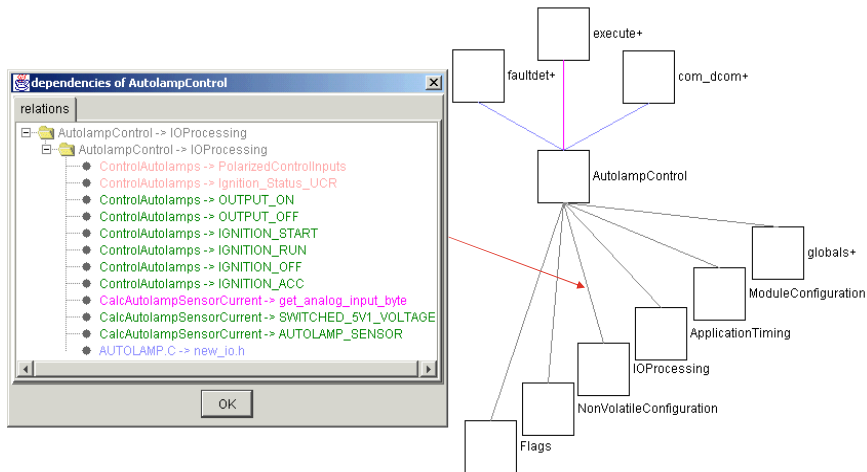
Identifying the relations



Cyclic Executive – function calls

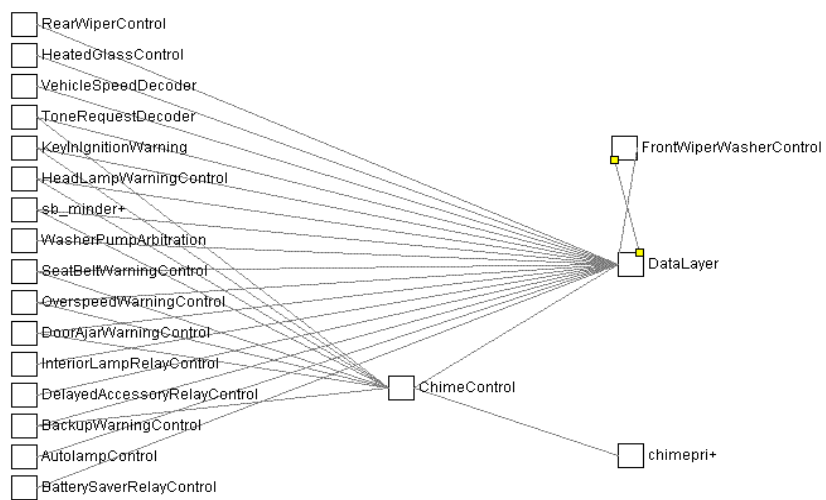


Investigating some component



ANU Lecture – Architecture-related Research

Feature interactions



ANU Lecture – Architecture-related Research

Obtain a high-level view of the architecture before starting reconstruction:

- helps to identify what to look for
- helps to identify what source information to extract (what information is architecturally relevant?)

Use “least effort” extraction

- can we use lightweight (lexical, perl) or heavyweight extraction (full-blown parsing)?

The reconstructor and architect need to work closely together when generating the abstractions

- try out different hypotheses and generate different views

It is important that the maintainers and developers are involved in the reconstruction process at an early stage

- they can help to identify what to look for

- Having an architecture is great for communication and analysis, but:
 - documented architectural decisions and implemented architectural decisions drift apart
- Architecture reconstruction is a *process* of rediscovering how architectural decisions were implemented
 - it's a lot of work, but worth it!

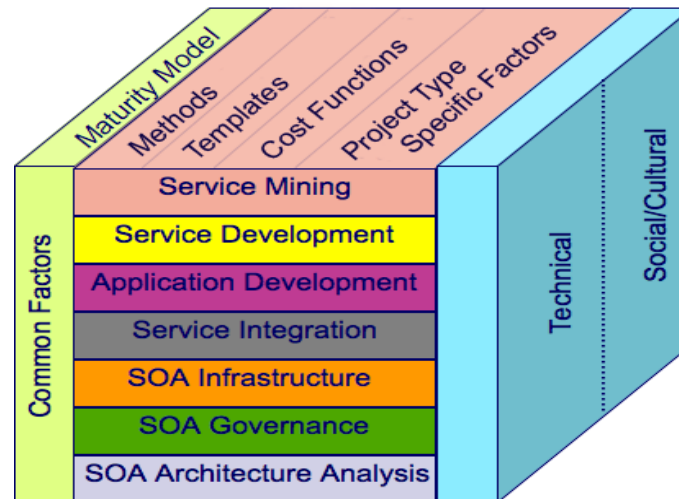
- Previously developed Service Mining and Reuse Technique (SMART)
 - Can use architecture reconstruction to understand existing systems
- Currently developing Scope, Cost and Effort Estimation Framework for SOA projects.

- In order to properly plan, budget and develop business cases for SOA projects (maybe an entire SOA Initiative) there is a need to properly scope, size, and cost such projects
- Currently there are few methods or techniques to determine the scope, effort and cost of these various types of SOA projects
- Looking at the published work on the topic of scoping, cost and effort estimation in SOA projects there are almost no papers published

- Does not seem to be a complete scoping, cost and effort estimation approach
- Most approaches focus on the technical work and not other dimensions
- Lack of details about what is actually involved in applying an approach
- Need specific approaches targeted to SOA
- Lack of transparency in how consulting firms and system integrators do scoping and estimate effort/cost
- No checklist to see if everything that should be part of an SOA project has been taken into consideration

Need a framework to bring all of this together

SMAT-AUS Framework – Scope, Cost and Effort Estimation Framework for SOA Projects



ANU Lecture – Architecture-related Research

41

Types of SOA Projects 1



- **Service Mining:** the identification and mining of services from existing (legacy) systems
- **Service Development:** the development of services from scratch
- **Application Development:** the development of applications from services
- **Service Integration:** the integration of common/shared services with existing (legacy) systems

ANU Lecture – Architecture-related Research

42

- **SOA Infrastructure:** the acquisition or development of an SOA infrastructure (including middleware, ESB, etc)
- **SOA Governance:** the development of governance policies and procedures (including establishing and monitoring Service Level Agreements)
- **SOA Architecture Analysis:** analysis of architectural alternatives – performance, scalability; security and other QoS requirements for SOA-based systems

An SOA initiative within an organisation will usually involve some combination of one or more of these types of projects

We envision that there are three main applications of the framework:

1. As a guide to organisations that want to undertake an SOA project – the Framework will allow them to better understand what needs to be done and how to do better cost and effort estimation
2. As a reference to check that a project has been scoped properly and that all of the costs and effort have been identified – especially in the case where a third party has produced a cost and effort estimate as in the case of a response to a RFP/RFT.
3. As an SOA capability assessment for organisations to determine how an organisation currently scopes and estimates its SOA projects

- The Framework is still being developed but major pieces are in place
 - SOA Integration project type analysed in details
 - Previous work on Service Mining project type could be incorporated
 - Details for SOA Infrastructure project type being finalised
 - Currently developing a series of models that provides an implementation of the framework
- Further work on social/cultural dimension is needed.
- Organisational SOA Maturity Model is still being developed
- Use of the Framework for Cost/Benefit Analysis