Architecting Flow in Software Engineering

Gail C. Murphy
University of British Columbia

@gail_murphy
Software engineering occurs at different scales
Software engineering occurs at different scales

**INDIVIDUAL COMPONENT**
E.G., BRAKE

**INTEGRATED COMPONENTS**
E.G., BRAKE & CRUISE

**CAR & ENVIRONMENT**
E.G., LANE DEPARTURE

**CAR-CAR INTERACTIONS**
E.G., SMART TRAFFIC
Software engineering involves multi-person multi-version development

— Brian Randell

[Parnas 2011]
Software engineering involves multi-person multi-version development

Tools to manage technical artifacts

And many more...
Software engineering involves **multi-person** multi-version development

*Tools to support social interaction*

![Bugzilla](https://via.placeholder.com/150)

![Discord](https://via.placeholder.com/150)

*And many more...*
Software engineering involves **multi-person multi-version** development

*Some integration between social and technical tools*

*But limited, more is possible*
Architecting Flow in Software Engineering

Software engineering is a socio-technical endeavour

Increasing focus on tool architectures and value streams can increase quality and productivity

S = Social Tool
T = Technical Tool
1. Socio-Technical Congruence
2. Information Flow
3. Tool Architecture
4. Architecting Flow
1. Socio-Technical Congruence
2. Information Flow
3. Tool Architecture
4. Architecting Flow
Socio-technical congruence

Congruence leads to improved productivity

Higher levels of congruence associated with lower levels of fault proneness

[Catalogo & Herbsleb, 2013]
Cataldo & Herbsleb studied two developments in depth

- **Project A** - Distributed system for a data storage product
  (114 developers, 5m lines of code, C & C++)

- **Project B** - Embedded system for automotive sector
  (380 developers, 8 sites, 7m lines of code, C)

finding that:

Higher-levels of socio-technical congruence are associated with better software quality. True for both projects, but with more benefit for Project B (more mature).

Higher-levels of socio-technical congruence associated with improvements in productivity. True for both projects, but with more benefit for Project A (less mature).

[Cataldo & Herbsleb, 2013]
Example 2: Integrated Systems

Brake software latest threat to Boeing 787

By Bill Rigby, Tim Hepher

FARNBOROUGH (Reuters) - Verifying software in the brake control system of Boeing’s BA.N 787 Dreamliner is the latest problem holding back the new plane’s first test flight, the troubled program’s chief said.

Example 3: Between Systems

What happens in ecosystems that are weakly inter-connected, like Java?

Red Dots are Java projects on GitHub.
The farther right, the more other technically dependent Java systems.

[Note] Palyart, Murphy & Masrani, 2017
Example 3: Between Systems

What happens in ecosystems that are weakly inter-connected, like Java?

This is a testing library (JUnit) with lots of systems using it.

[R: Ratio of user repositories having a social link]

[Palyart, Murphy & Masrani, 2017]
Example 3: Between Systems

What happens in ecosystems that are weakly inter-connected, like Java?

The vertical axis provides an indication of how many of the using systems have a social dependence.

Reasonable number of using systems and high-level of social dependencies from those systems.

Lots of users, less social dependencies from those systems.

[Palyart, Murphy & Masrani, 2017]
## Architecting Flow

<table>
<thead>
<tr>
<th>Scale</th>
<th>Why (examples)</th>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within system</td>
<td>Increased productivity and quality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrated system</td>
<td>Avoid integration failures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between systems</td>
<td>Improve planning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Architecting Flow

<table>
<thead>
<tr>
<th>Scale</th>
<th>Why (examples)</th>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within system</td>
<td>Increased productivity and quality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrated system</td>
<td>Avoid integration failures</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Focus will be on “within system” and “between systems” for remainder of talk.
Technical dependencies between modules

A → C

B

Coordination needs between teams

Team for A

Team for B

Team for C

Information flow

Information flow related to technical dependence
Is related to software architecture

Software architecture specification, evolution, recovery, practices, etc. are well-established
Technical dependencies between modules

A → C
→ B

Information flow

Information flow to support coordination needs is less established

Focus in this talk will be on information flow related to coordination

Coordination needs between teams

Team for A

Team for B

Team for C
Types of information flowing through a software development

- Features
- Defects
- Debts
- Risks

Four types of Flow Items
– Mik Kersten, 2018
A “Within System” Conceptual Example

An automotive company building a hybrid or electric vehicle with regenerative braking
**Requirement:** Provide regenerative braking

**Task:** Design algorithm for regenerative braking

**Task:** Simulate algorithm for regenerative braking

**Task:** Implement algorithm for regenerative braking

**Task:** Test algorithm for regenerative braking

**Task:** Performance test algorithm for regenerative braking

**Task:** Gain input from brake sensors

**Task:** Reverse engine

**Task:** Route electricity from motor to batteries

**Information flow items related to a Feature**

Etc. etc. etc.
<table>
<thead>
<tr>
<th><strong>Requirements Team</strong></th>
<th><strong>Requirement</strong>: Provide regenerative braking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Team</strong></td>
<td><strong>Task</strong>: Design algorithm for regenerative braking</td>
</tr>
<tr>
<td><strong>Dev Team #1</strong></td>
<td><strong>Task</strong>: Simulate algorithm for regenerative braking</td>
</tr>
<tr>
<td><strong>Dev Team #2</strong></td>
<td><strong>Task</strong>: Implement algorithm for regenerative braking</td>
</tr>
<tr>
<td><strong>Test Team</strong></td>
<td><strong>Task</strong>: Test algorithm for regenerative braking</td>
</tr>
<tr>
<td></td>
<td><strong>Task</strong>: Performance test algorithm for regenerative braking</td>
</tr>
</tbody>
</table>

Information flow items related to a *Feature*

Etc. etc. etc.
Information flow items related to a Feature

Requirements Team → Design Team → Dev Team #1 → Dev Team #2 → Test Team

Etc. etc. etc.
Information flow items related to a **Defect**

- Requirements Team
- Design Team
- Dev Team #1
- Dev Team #2
- Test Team
- Defect Team

**Defect: Problems with anti-lock braking**

*Image not copyable*
Technical: Algorithm Spec
Social: Discussion about Spec

Information flow items related to a Feature and **Defect**

Etc. etc. etc.

Technical: Environment conditions for defect
Social: Discussion to gather more environmental info
An “Integrated System” Conceptual Example

An airline manufacturer contracting the braking system for the plane
Information flow items

Integrator

Feature:
Provide braking component

Supplier
Information flow items

**Feature:** Provide braking component

**Technical:** component

**Social:** interface discussion

More limited content on flow Items between integrator and supplier

Supplier

- Requirements Team
- Development Team
- Quality Assurance Team
<table>
<thead>
<tr>
<th>Scale</th>
<th>Why (examples)</th>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within system</td>
<td>Increased productivity and quality</td>
<td>Features, Defects, Risks and Debts</td>
<td></td>
</tr>
<tr>
<td>Integrated system</td>
<td>Avoid integration failures</td>
<td>Likely contractually limited to features and defects</td>
<td></td>
</tr>
</tbody>
</table>
1. Socio-Technical Congruence
2. Information Flow
3. Tool Architecture
4. Architecting Flow
Modularization of software to reduce team interdependencies degrades over time.

Teams interchange a variety of technical and social information during development.

Information flow is constant, complex and changing.

Organization and software structure will not suffice.
Tools can help track information

Requirements Team
- Requirements

Design Team
- Features

Dev Team #1
- Tasks
- Defects

Dev Team #2
- Tasks
- Defects

Test Team
- Test
Tool architectures with an integration substrate can help information flow.
An example tool architecture

With thanks to Tasktop Technologies Inc. for the diagram.
Diagram may not be copied.
Tool architectures vary

With thanks to Tasktop Technologies Inc. for the diagram.
Diagram may not be copied.
Tool architectures vary

An integration substrate (product) can enable flow across the connectors between tools

With thanks to Tasktop Technologies Inc. for the diagram. Diagram may not be copied.
What about integrated system development?

A deep relationship between integrator and supplier may see information flow between tools.

A less trusted relationship may see the Integrator limited to sending defects to the supplier.
# Architecting Flow

<table>
<thead>
<tr>
<th>Scale</th>
<th>Why (examples)</th>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within system</td>
<td>Increased productivity and quality</td>
<td>Features, Defects, Risks and Debts</td>
<td>Tool architecture</td>
</tr>
<tr>
<td>Integrated system</td>
<td>Avoid integration failures</td>
<td>Likely contractually limited to features and defects</td>
<td>Tool architecture</td>
</tr>
</tbody>
</table>
1. Socio-Technical Congruence
2. Information Flow
3. Tool Architecture
4. Architecting Flow
Information flow is technical and social information and is about features, defects, risks and debts.

Tool architectures supported by an integration substrate enable information flow.

Socio-technical congruence can increase quality and improve productivity.

But what information flows should be supported?
Information flow related to value streams needs to be supported.

A value stream takes a product or service from beginning through to a customer (who may be internal or external).
How does a value stream relate to software architecture?
Tracking Information Flow for Value Streams
Supporting software engineering at different scales

<table>
<thead>
<tr>
<th>Scale</th>
<th>Tracking Information Flow for Value Steams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within system</td>
<td>Commercial products</td>
</tr>
<tr>
<td>Integrated system</td>
<td>Commercial products (depending on trust relationship)</td>
</tr>
</tbody>
</table>
# Architecting Flow

<table>
<thead>
<tr>
<th>Scale</th>
<th>Why (examples)</th>
<th>What</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within system</td>
<td>Increased productivity and quality</td>
<td>Features, Defects, Risks and Debts</td>
<td>Value streams and tool architecture</td>
</tr>
<tr>
<td>Integrated system</td>
<td>Avoid integration failures</td>
<td>Likely contractually limited to features and defects</td>
<td>Value streams and tool architecture</td>
</tr>
</tbody>
</table>
Many open questions

Conway’s law (adage) 1967

Modular software architecture to enable parallel work and enable change (1970s)

but...

In practice, changes crosscut software architecture, attention is being placed on (often crosscutting) value streams and developers view their teams as fluid

so...

How should software be organized? How should teams be organized? What metrics should be tracked?
Socio-Technical Congruence

Information Flow

Tool Architecture

Architecting Flow

Summary
Thanks

To the many undergraduate and graduate students, post-docs, and colleagues who have contributed to how I think about software engineering and research.

To Mik Kersten and my colleagues at Tasktop for the many great discussions, collaborations and insights.

To the organizers of ESI 2022 for the kind invitation to present this talk.
Socio-technical congruence

Information flow

Tool architectures

Value streams
Architecting Flow in Software Engineering

Software engineering is a socio-technical endeavour

Increasing focus on tool architectures and value streams can increase quality and productivity

@gail_murphy

