Knowledge Sharing in Software Development

Hans van Vliet, Antony Tang

VU University Amsterdam

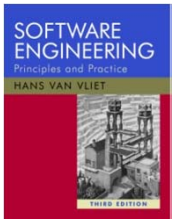
QSIC 2010

July 14, 2010

vrije Universiteit amsterdam

My personal history

1967	computer operator, programmer
1973-1978	MSc Mathematics/CS
1979	PhD, ALGOL 68
1986	Professor Software Engineering, VU University
1983	Software Engineering textbook (2000, 2008)
2008	Journal of Systems and Software (EiC)
1996-	Research Software Architecture (ALMA, GRIFFIN, Stephenson)




Fiore

Main hypothesis

Better knowledge sharing

$\Rightarrow$

Better quality

Summary

- Software people are not inclined to share knowledge
- There is a lot of knowledge in the artefacts
- Feedback based on continuous/regular mining of artefacts

The Griffin Consortium

CIBIT
a DNV company

PHILIPS

RuG

ASTRON

vrije Universiteit amsterdam

Getronics
PinkRoccade

LogicaCMG

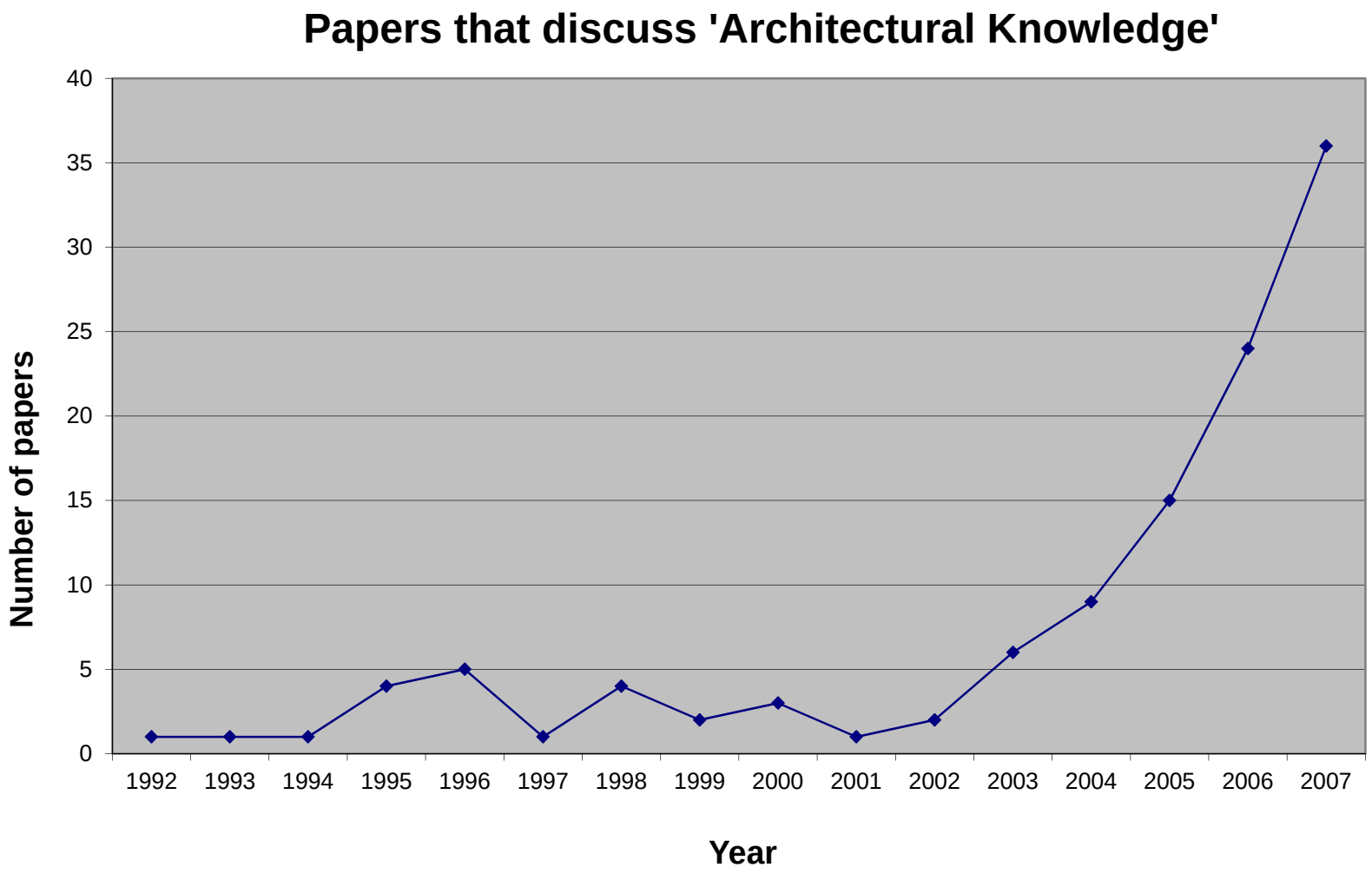
CHESS

Software architecture

- Software architecture = components + connectors
- Software architecture = set of design decisions
- Software architecture knowledge = solution (components + connectors) + why this solution (design decisions + rationale)

Architecture of WWW – design decisions (according to Taylor/Medvidovic/Dashevy book)

- Web is a collection of resources with unique names (URL)
- Each resource denotes some information
- URI can be used to determine the identity of a machine
- Communication is initiated by clients
- Resources can be manipulated through their representation
- All communication goes through their representation, with commands like GET, POST, …
- Interactions are stateless

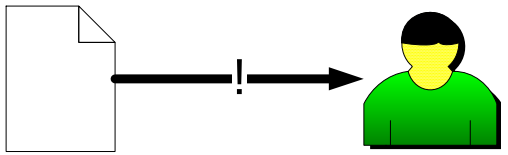


Research themes Griffin

Sharing

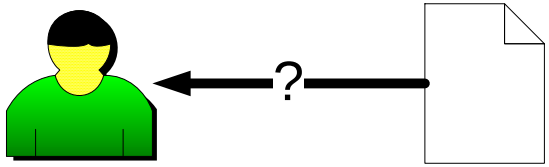
Improve the way AK is shared within and between organizations

Compliance



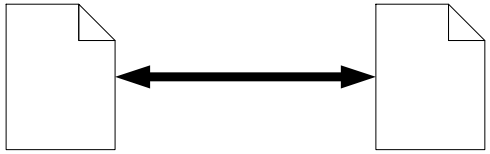
Support alignment to common architectural rules

Discovery



Enhance the findability of relevant Architectural Knowledge

Traceability



Enable effective navigation through an organization's body of AK

Theoretical Framework -
What we thought architects do

1. **Communicate**
   - Inform colleagues & customers, explain things, discuss, etc.
2. **Make decisions**
   - Weigh pros & cons, consider alternatives and make tradeoffs
3. **Assess quality**
   - Convince stakeholders, ensure QAs, assess architectures
4. **Document**
   - Describe architectural solutions, (re)use templates, etc.
5. **Acquire knowledge**
   - By training, reading, attending conferences, using the Internet

Theoretical Framework –
Support we thought architects would need

1. **Decision management support**
   - Overviews, codification templates, repositories, traceability
2. **Efficient search mechanisms**
   - Search for architectural guidelines and rules, decisions, or relevant project information
3. **Community building support**
   - Who is doing/knowing what? Ease collaboration, notifications
4. **Intelligent advice**
   - Pro-active feedback, specific suggestions, overviews
5. **Knowledge management support**
   - Automatic retrieval of AK, annotation support, central management of knowledge entities

Survey analysis – main findings 1/3

- **What architects do**
  - Decision making is their most important activity
  - Documenting the results is their least important activity
  - More experienced architects spend more time in documenting and in quality assessment

- **Discussion**
  - Lack of a defined visible process does not induce documenting
  - Architects have a lot of tacit knowledge that is hard to codify
  - Documenting results could make architects redundant
  - The prevalent mindset of architects is focused on ‘to create and communicate’ rather than ‘to review and maintain’ an architecture

Survey analysis – main findings 2/3

- **What architects need**
  - Architects mostly need effective ways of searching AK and support for decision management
  - Architects are not that fond of knowledge management or intelligent support

- **Discussion**
  - There is a lack of balance between production and consumption of knowledge (architects really need previously stored knowledge, but refrain from documenting it…)
  - Architects wish to remain in control; no tool, method or colleague can take over their role
  - This might also explain why community building support was ranked rather low

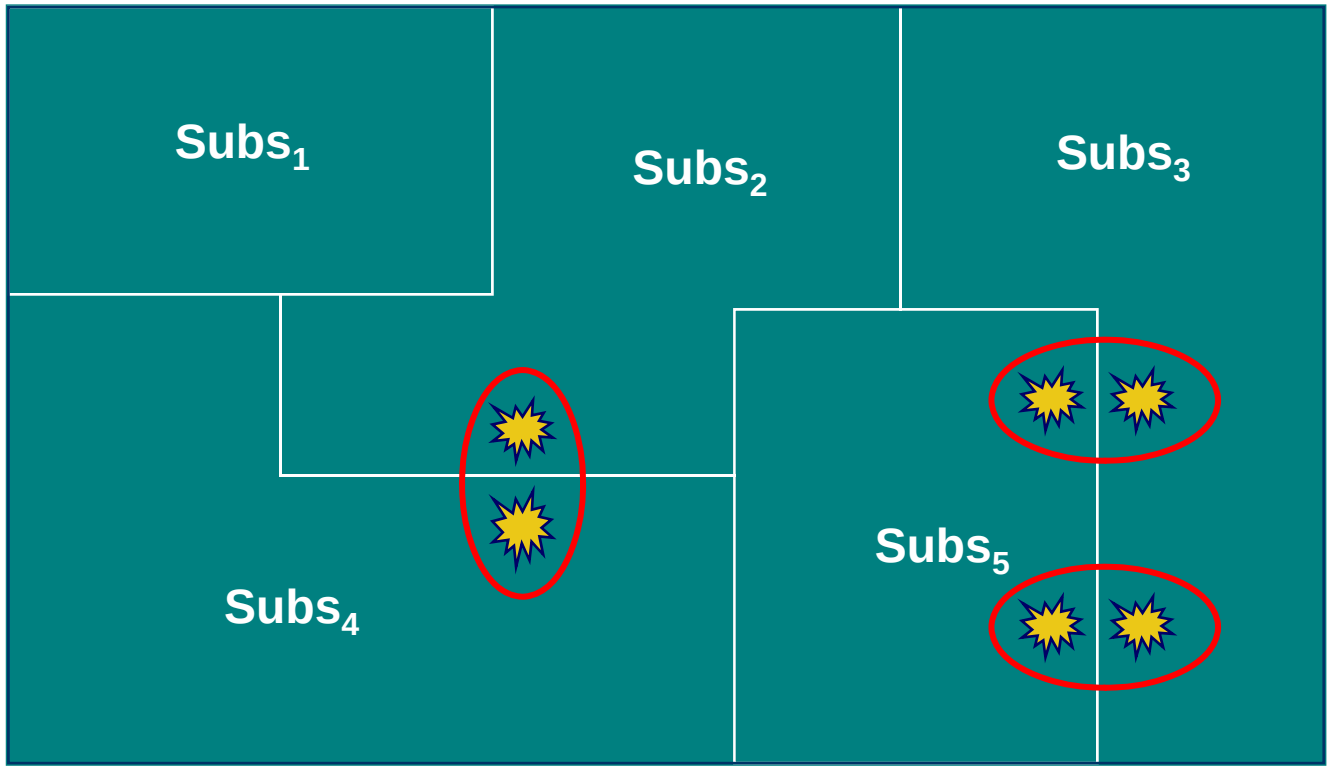
Meet the ‘lonesome architect’ (WICSA 2009)

- Someone who takes all major design decisions
- Someone who cares less about documenting and sharing AK
- Someone not that interested in automated or intelligent support
- Someone who demands effective ways of searching AK

Summary

- Software people are not inclined to share knowledge
- There is a lot of knowledge in the artefacts
- Feedback based on continuous/regular mining of artefacts

How (un)evolvable is this architecture?



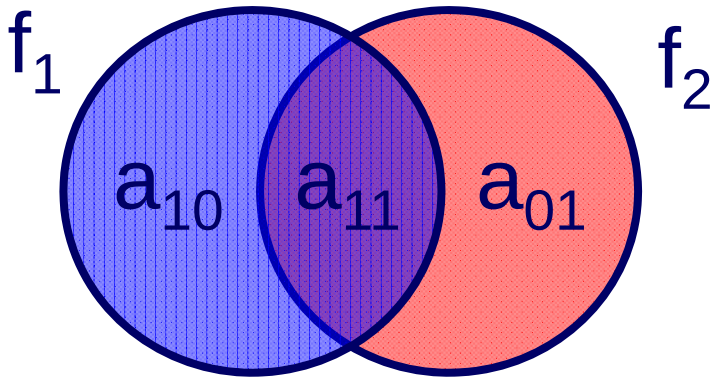
Context

Magnetic Resonance (MR)

- **hundreds of developers**
- **decades of development**
- **millions LOC**
- **developed in C, C++, C#, …**

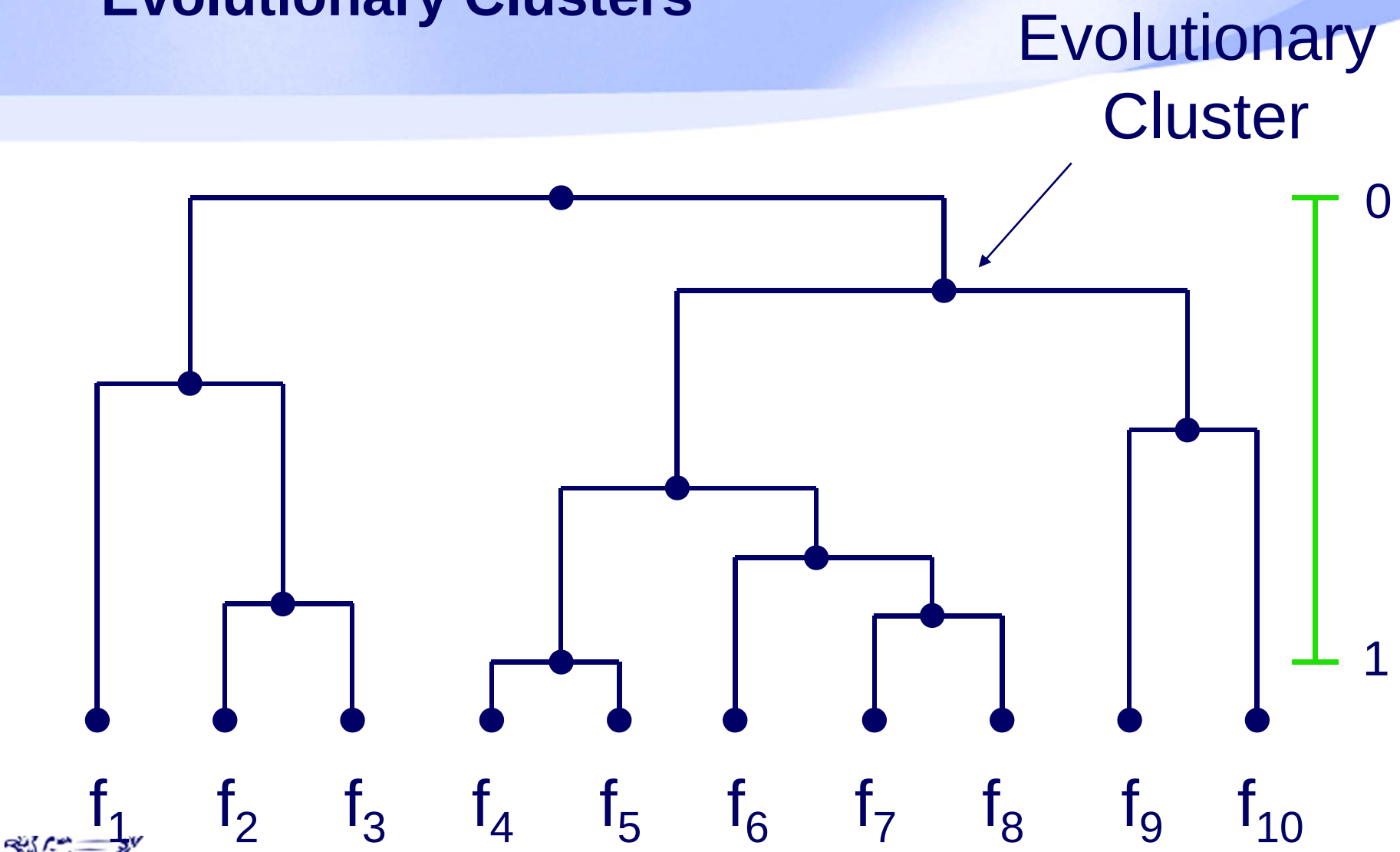
Modification Similarity of Files

- Computed using the Jaccard similarity metric

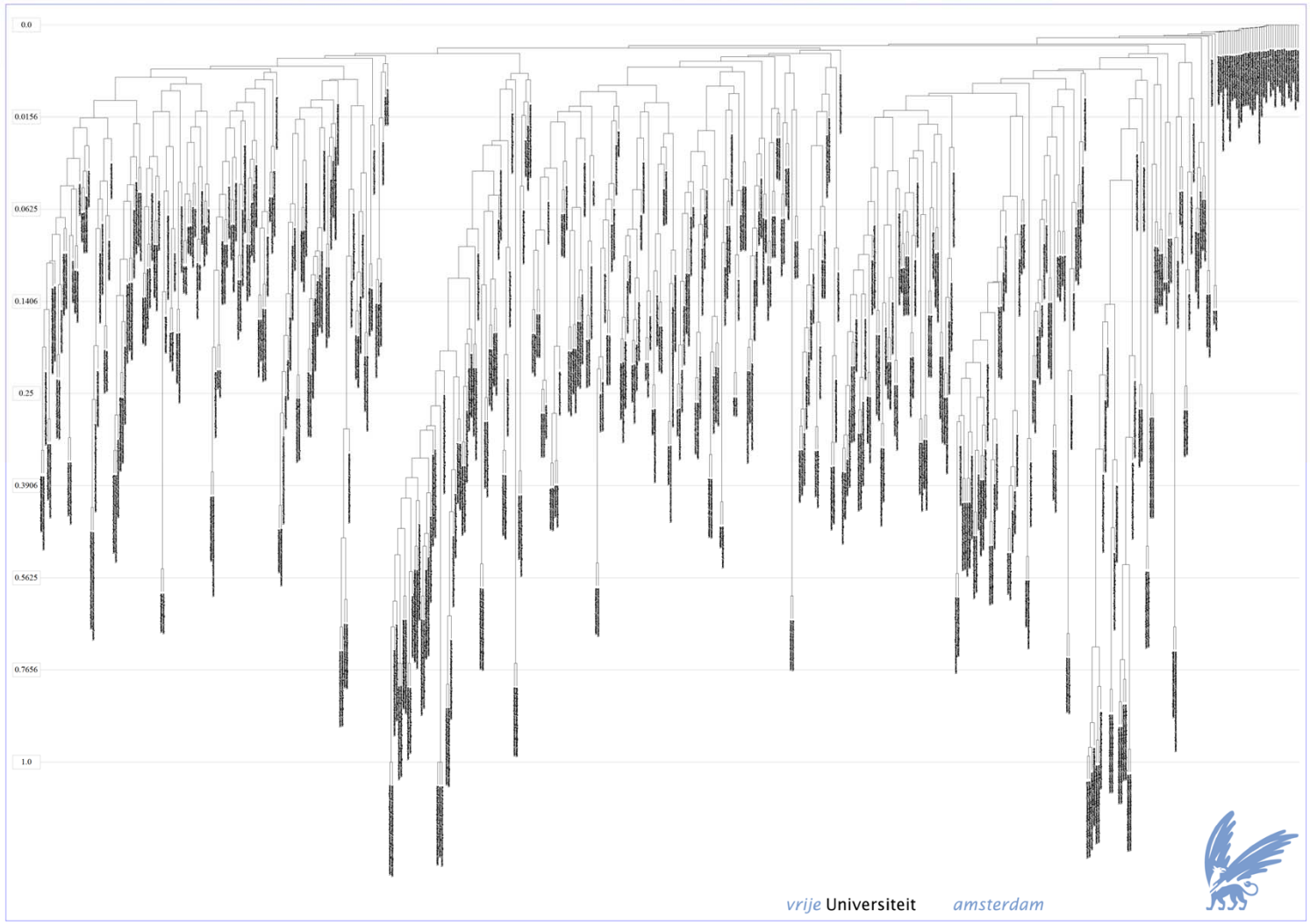


$$S(f_1, f_2) = a_{11}/(a_{11} + a_{10} + a_{01})$$

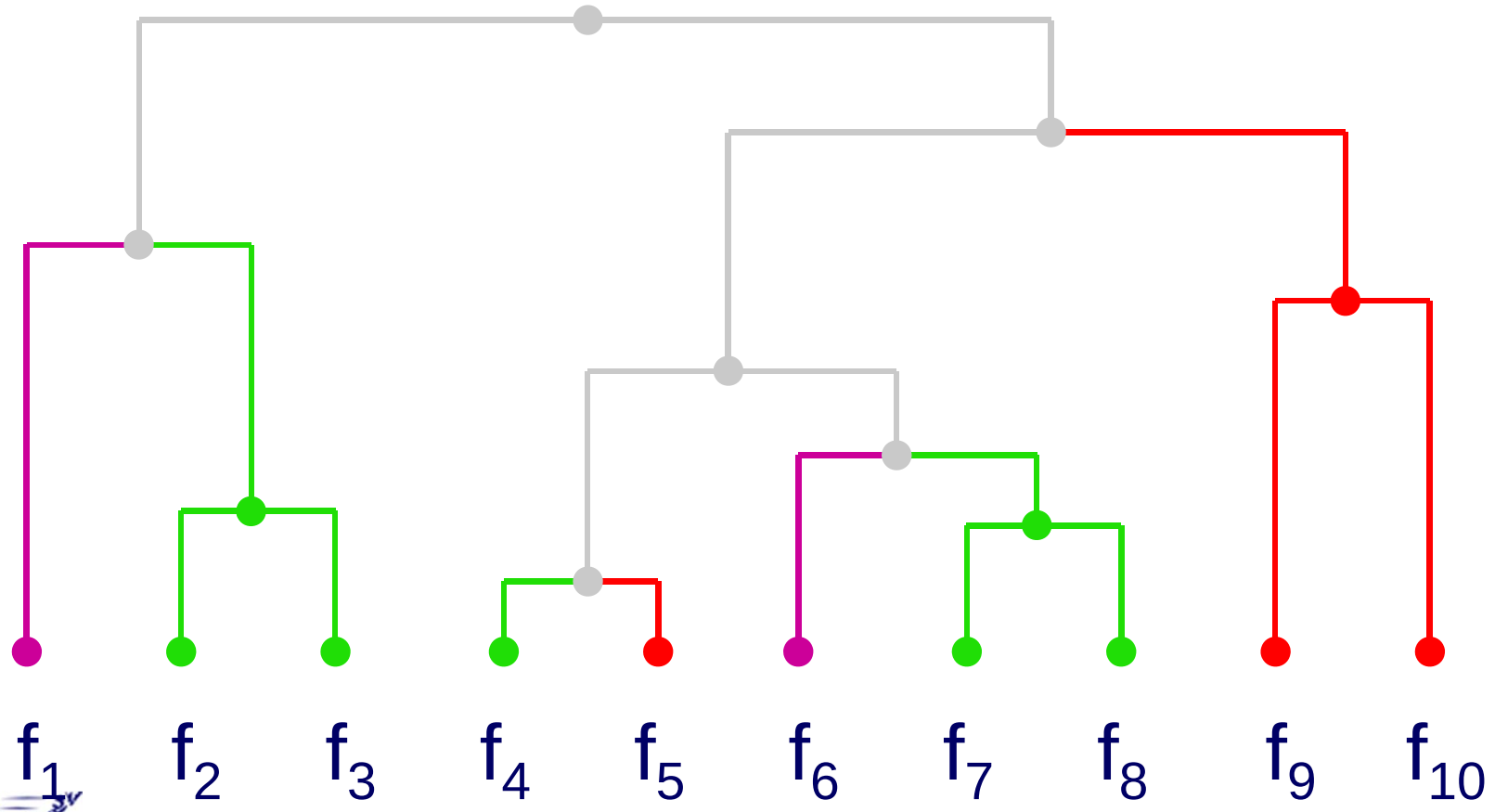
Evolutionary Clusters

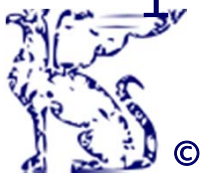


Example cluster hierarchy

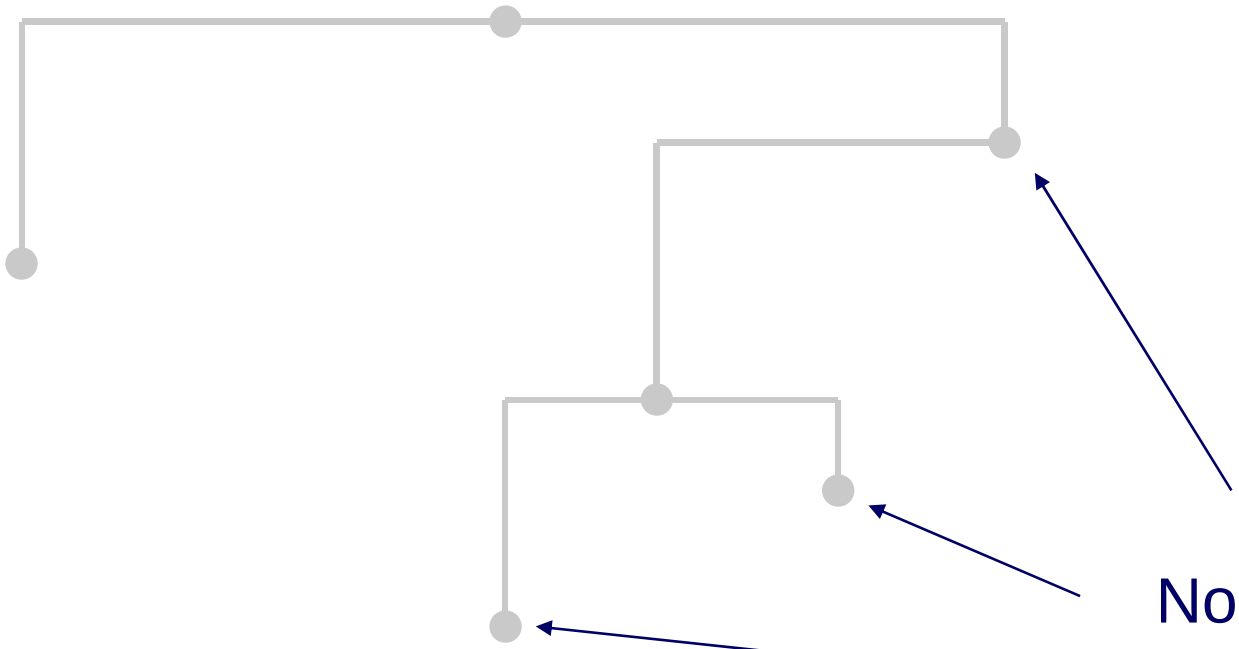


Filtering the hierarchy





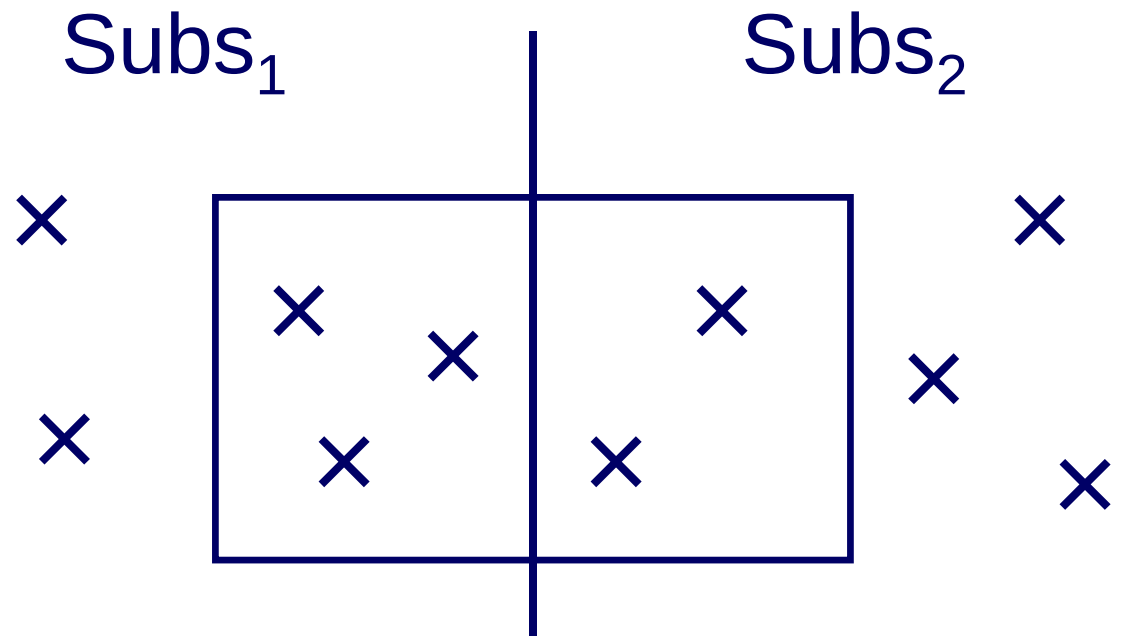
Filtered hierarchy and hot spots



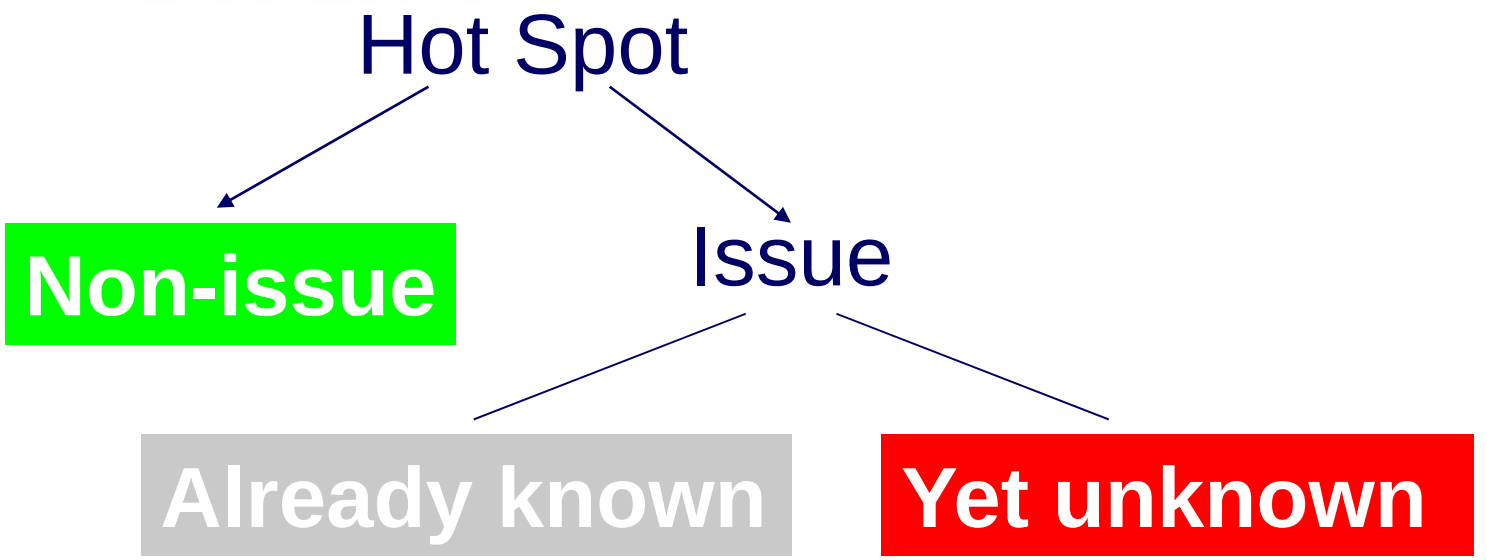
Non-trivial Evolutionary Clusters

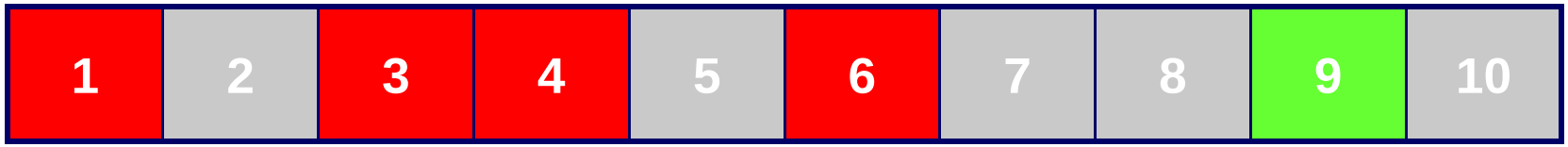
Hot Spots

- **Hot Spots are software entities (files or building blocks) changing together frequently over the borders of different subsystems.**



The Top 10 Hot Spots





Relevant properties of a hot spot

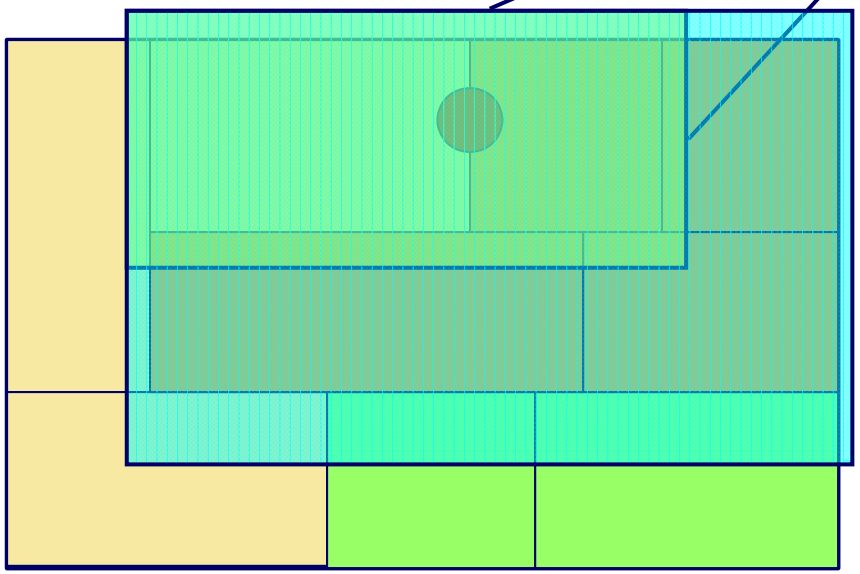
- Support (Jaccard value)
- Size
- Different subsystems/development groups/ development sites/ hardware…
- When co-evolved first/last, tendency
- …

The characterization

HS 7		Distributions					Borders Crossed		
		Confidence	Support	First co-evolution	Last co-evolution	CE Tend	DEV	IND	HWP
Size: 15	MAX	**0.1769**	**106**	**16 Oct 2004**	**07 Oct 2006**	**0.0593**	OK	!!!	OK
	MIN	**0.0615**	**35**	**04 Apr 2005**	**16 Dec 2006**	**-0.2405**			
	AVG	**0.1050**	**61**	**19 Feb 2005**	**28 Nov 2006**	**-0.1180**			
	SIG	**0.0279**	**16**	**74**	**19**	**0.0590**			

Borders Crossed 1

- Development Groups
- Independent Release Group
- Hardware pieces



Borders Crossed		
DEV	IND	HWP
OK	!!!	OK

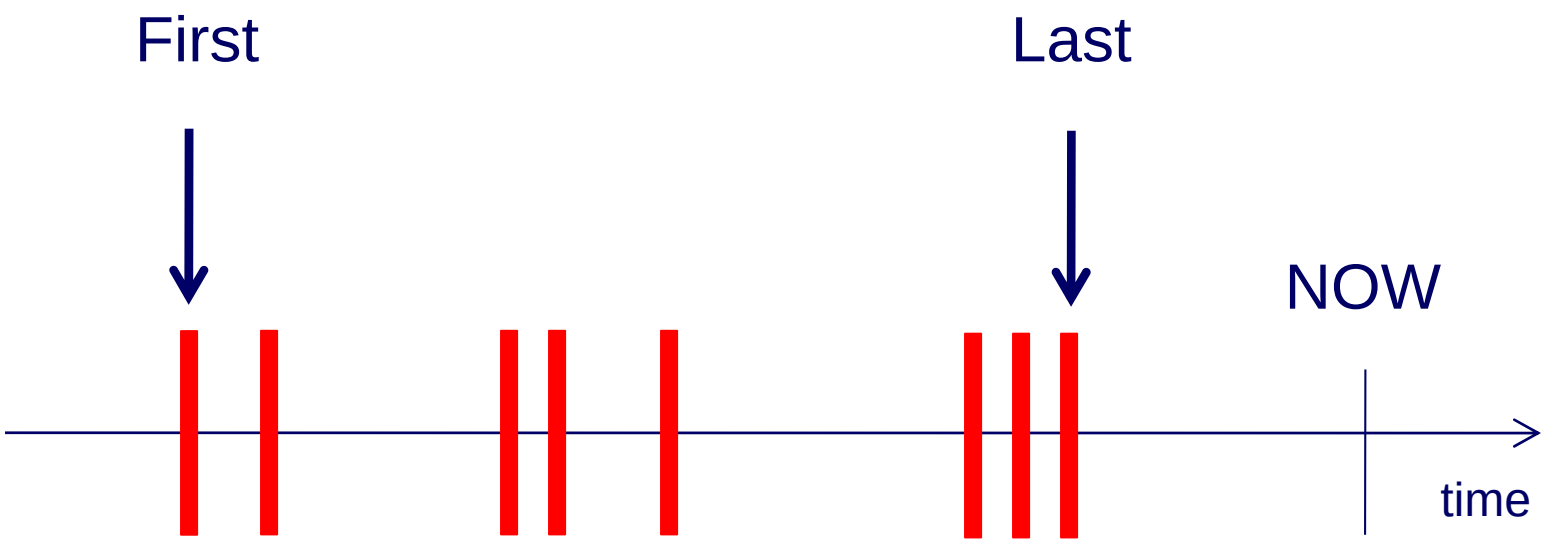
The Characterization

HS 7		Distributions					Borders Crossed		
		Confidence	Support	First co-evolution	Last co-evolution	CE Tend.	DEV	IND	HWP
Size: 15	MAX	**0.1769**	**106**	**16 Oct 2004**	**07 Oct 2006**	**0.0593**	OK	!!!	OK
	MIN	**0.0615**	**35**	**04 Apr 2005**	**16 Dec 2006**	**-0.2408**			
	AVG	**0.1050**	**61**	**19 Feb 2005**	**28 Nov 2006**	**-0.1180**			
	STG	**0.0279**	**16**	**74**	**19**	**0.0590**			

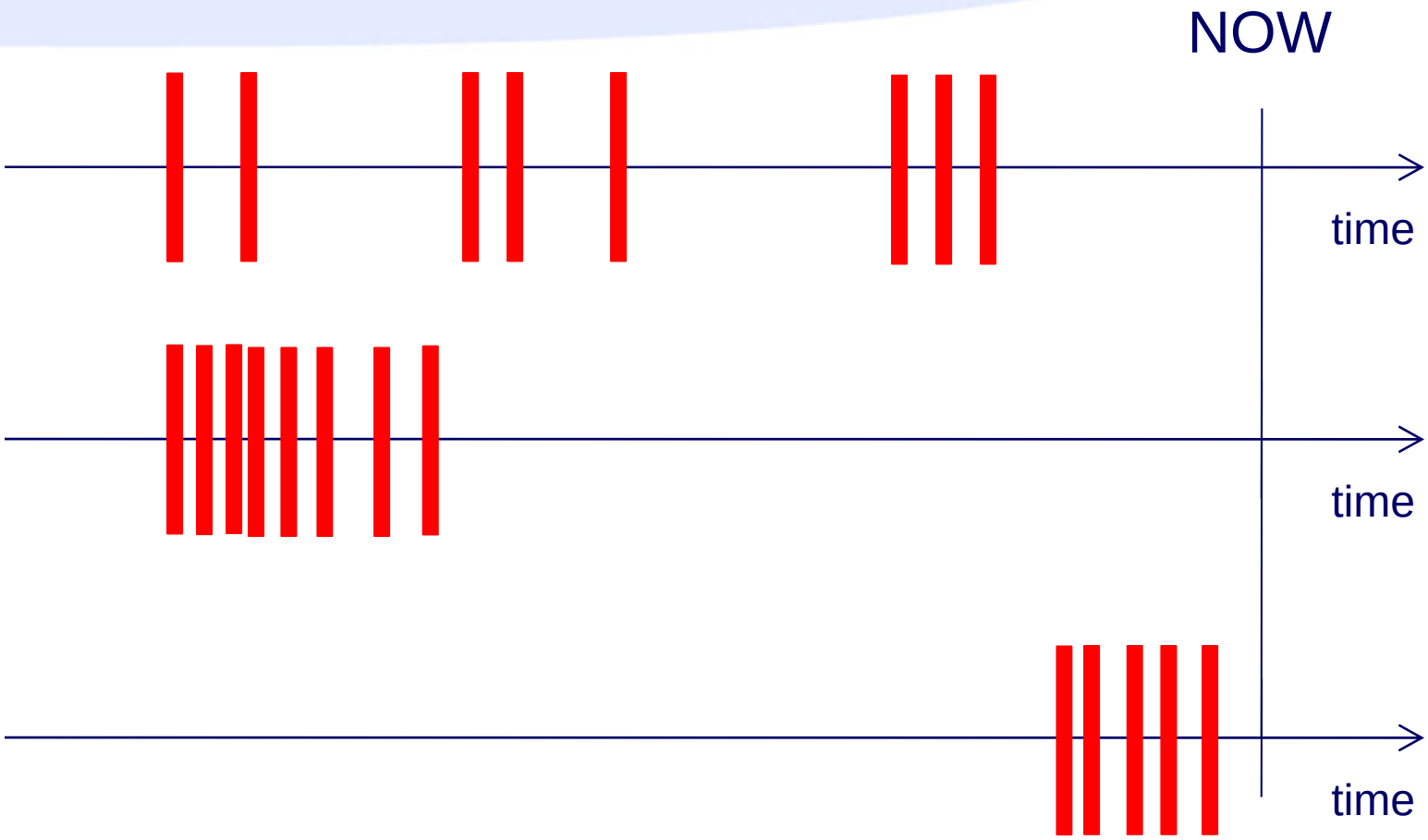
The characterization

HS 7		Distributions					Borders Crossed		
		Confidence	Support	First co-evolution	Last co-evolution	CE Tend.	DEV	IND	HWP
Size: 15	MAX	**0.1769**	**106**	**16 Oct 2004**	**07 Oct 2006**	**0.0593**	OK	!!!	OK
	MIN	**0.0615**	**35**	**04 Apr 2005**	**16 Dec 2006**	**-0.2408**			
	AVG	**0.1050**	**61**	**09 Feb 2005**	**28 Nov 2006**	**-0.1180**			
	SIG	**0.0379**	**16**	**74**	**19**	**0.0590**			

Distribution of Co-evolutions 1



Distribution of Co-evolutions 2



The characterization

HS 7		Distributions					Borders Crossed		
		Confidence	Support	First co-evolution	Last co-evolution	CE Tend.	DEV	IND	HWP
Size: 15	MAX	**0.1769**	**106**	**16 Oct 2004**	**07 Oct 2006**	**0.0593**	OK	!!!	OK
	MIN	**0.0615**	**35**	**04 Apr 2005**	**16 Dec 2006**	**-0.2408**			
	AVG	**0.1050**	**61**	**19 Feb 2005**	**28 Nov 2006**	**-0.1180**			
	SIG	**0.0279**	**16**	**74**	**19**	**0.0590**			

Which One Is More Important?

HS 7		Distributions					Borders Crossed		
		Confidence	Support	First co-evolution	Last co-evolution	CE Tendency	DEV	IND	HWP
Size: 15	MAX	**0.1769**	**106**	**16 Oct 2004**	**07 Oct 2006**	**0.0593**	OK	!!!	OK
	MIN	**0.0615**	**35**	**04 Apr 2005**	**16 Dec 2006**	**-0.2408**			
	AVG	**0.1050**	**61**	**19 Feb 2005**	**28 Nov 2006**	**-0.1180**			
	SIG	**0.0279**	**16**	**74**	**19**	**0.0590**			

HS 8		Distributions					Borders Crossed		
		Confidence	Support	First co-evolution	Last co-evolution	CE Tendency	DEV	IND	HWP
Size: 3	MAX	**0.1226**	**26**	**14 Apr 2000**	**19 Jun 2006**	**-0.3681**	!!!	!!!	!!!
	MIN	**0.0869**	**24**	**14 Apr 2000**	**19 Jun 2006**	**-0.3704**			
	AVG	**0.1047**	**25**	**14 Apr 2000**	**19 Jun 2006**	**-0.3693**			
	SIG	**0.0017**	**1**	**0**	**0**	**0.0013**			

Which One is More Important?

- The answer is context dependent
- Priorities have to be determined according to the interests of architects / developers $\Rightarrow$ the list of hot spots is pruned by executing queries on them (like “I am only interested in hot spots that cross development sites, where most changes occurred recently”)

Summary

- Software people are not inclined to share knowledge
- There is a lot of knowledge in the artefacts
- Feedback based on continuous/regular mining of artefacts

A printer product line

Characteristics

- Product lines for both wide format printing and document printing (in total: 7)
- Many products under development/in operation
- Controller software: ~250 people
- 3 sites, in 3 countries (& legal relations differ)

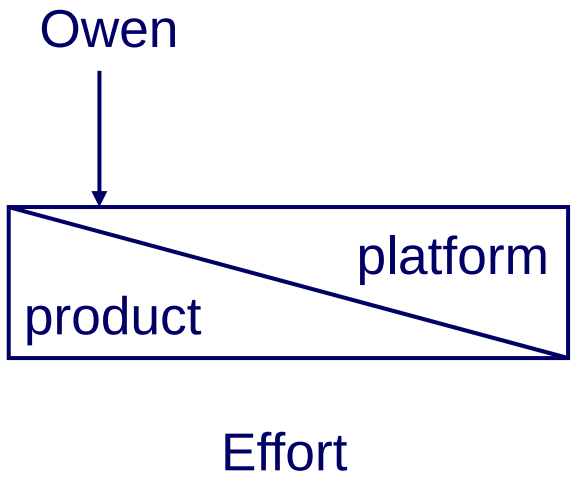
Tensions

- **How much to document?**
  - Everything upfront? – classic approach
  - Nothing? – pure agile
  - Somewhere in between, just-in-time, just-enough

- **How much to plan?**
  - Grand upfront design? – enterprise architecture/product line architecture document
  - Let all flowers bloom
  - Somewhere in between? – “Owen” model

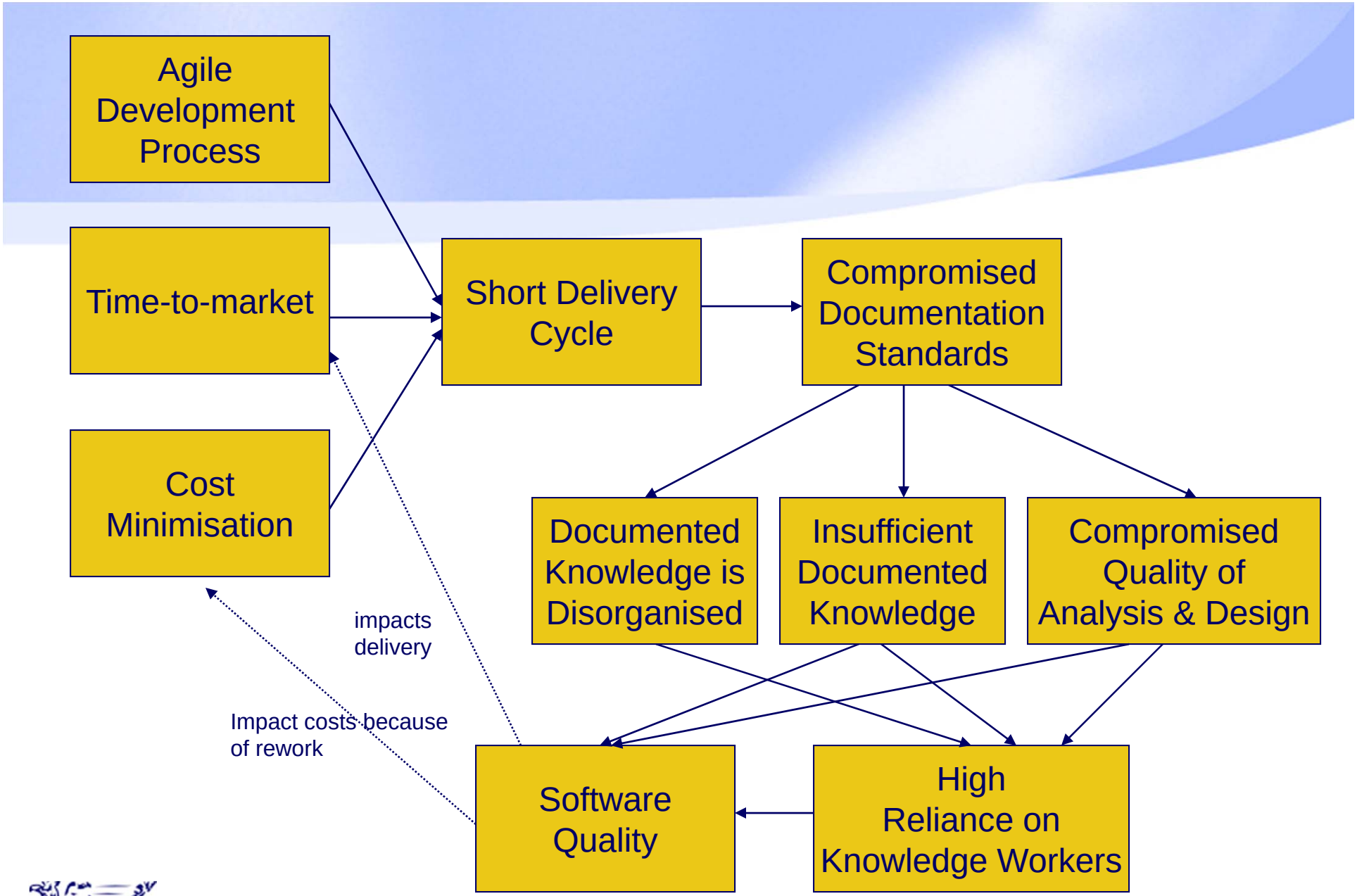
Owen model

Product Focus
- Platform dev by product teams
- Small platform team
- Co-operative work



Platform Focus
- Large platform team
- Many small product teams
- Hierarchical work

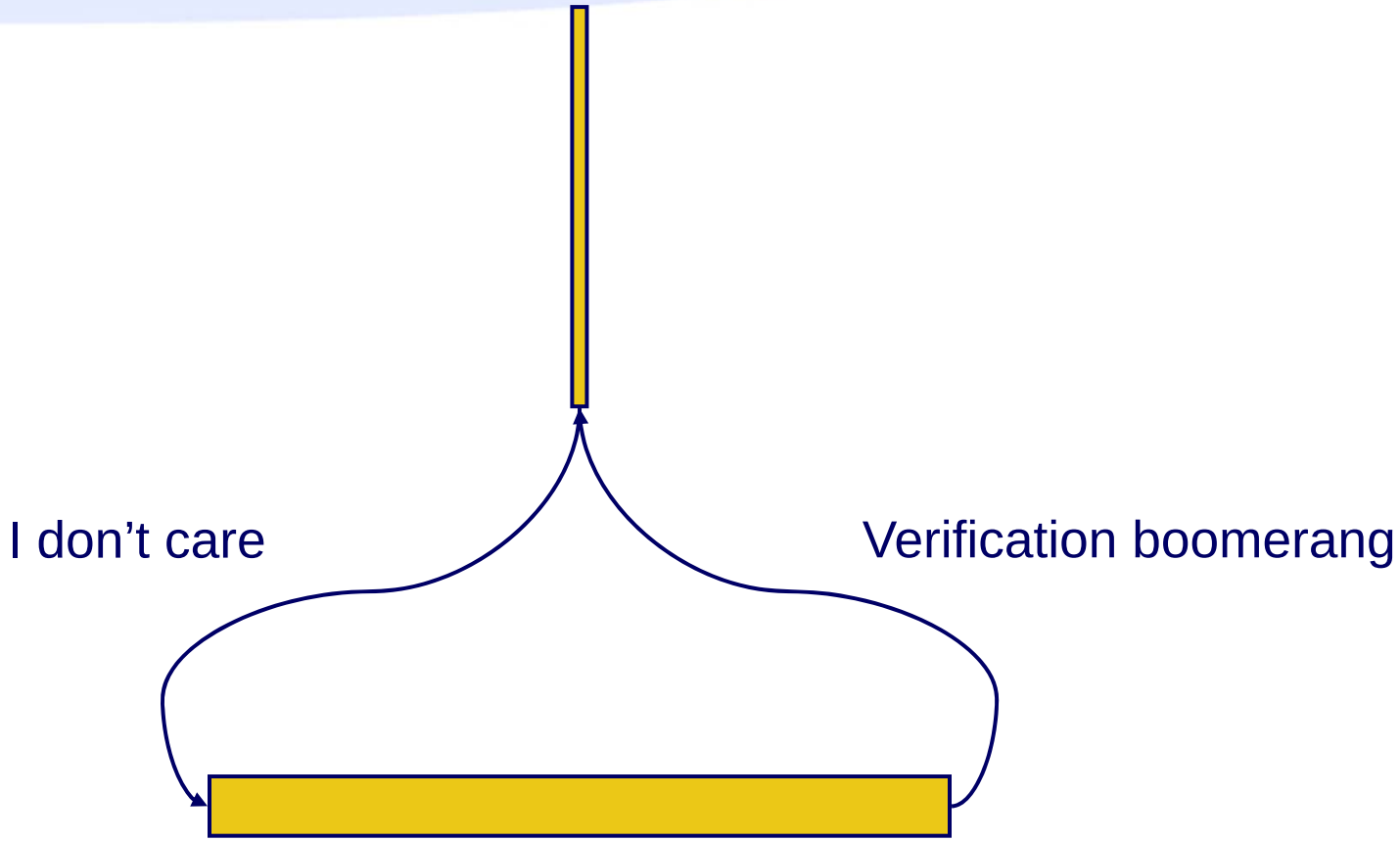
Toft et al, A cooperative model for cross-divisional product development, SPLC1



Example issues we investigate

- How much should the architect tell
- How far ahead should an architect plan
- Knowledge sharing issues between sites

How much should the architect tell



How much should the architect tell

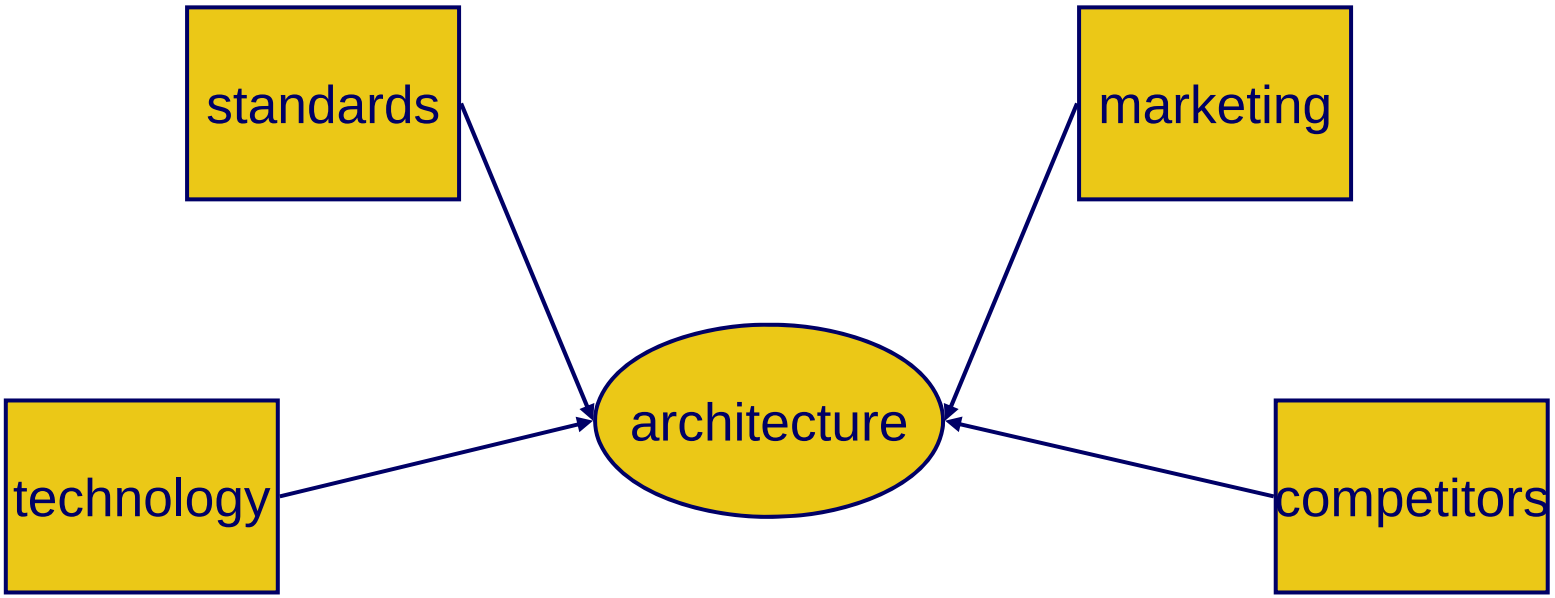
- **Investigate database with >400 architecture-related bug reports**
- **Classify causes:**
  - Quality of architecture documents
  - Personnel volatility (multisite, agile)
  - (Assumptions about) knowledge of developers (domain, limitations, where to look)

Dynamic feedback by mining bugs

advice

bugs, level of detail, …

How far ahead should an architect plan



How far ahead should an architect plan

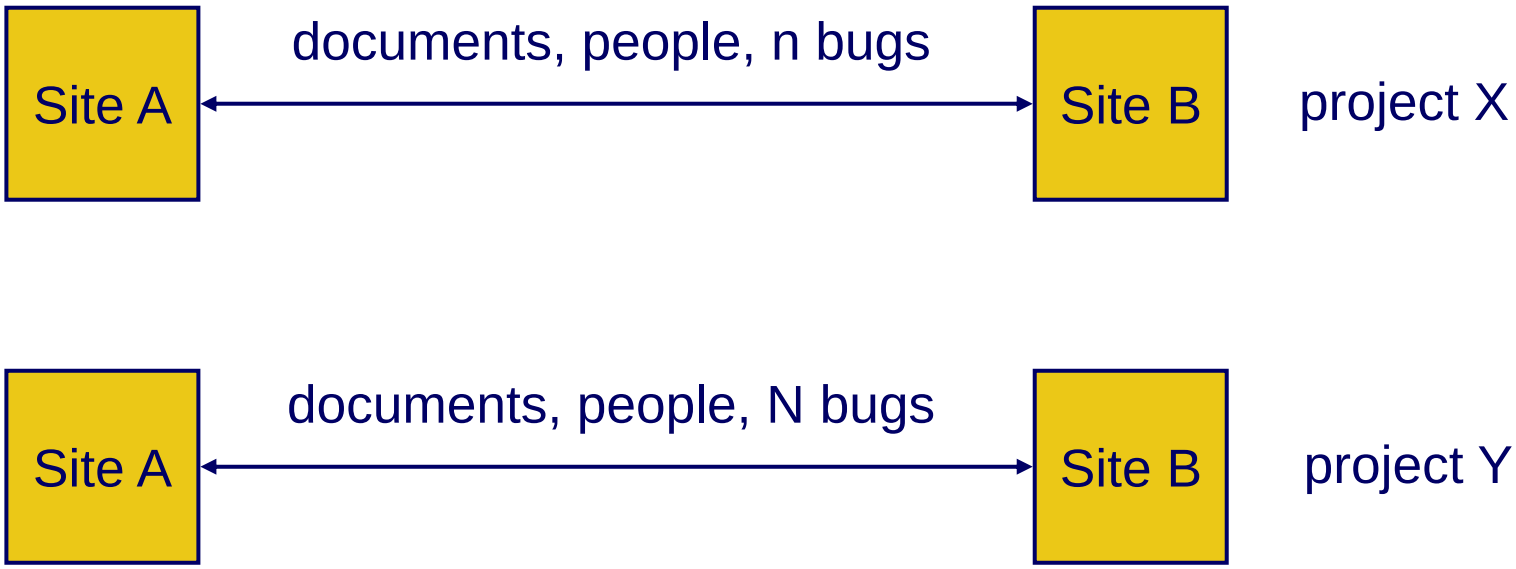
- Short-term, reactive planning (quick-fix) vs
- Long-term, architecture-guided planning
- Time to market/cost/flexibility

Dynamic feedback about roadmaps used

advice

roadmaps, time & cost, …

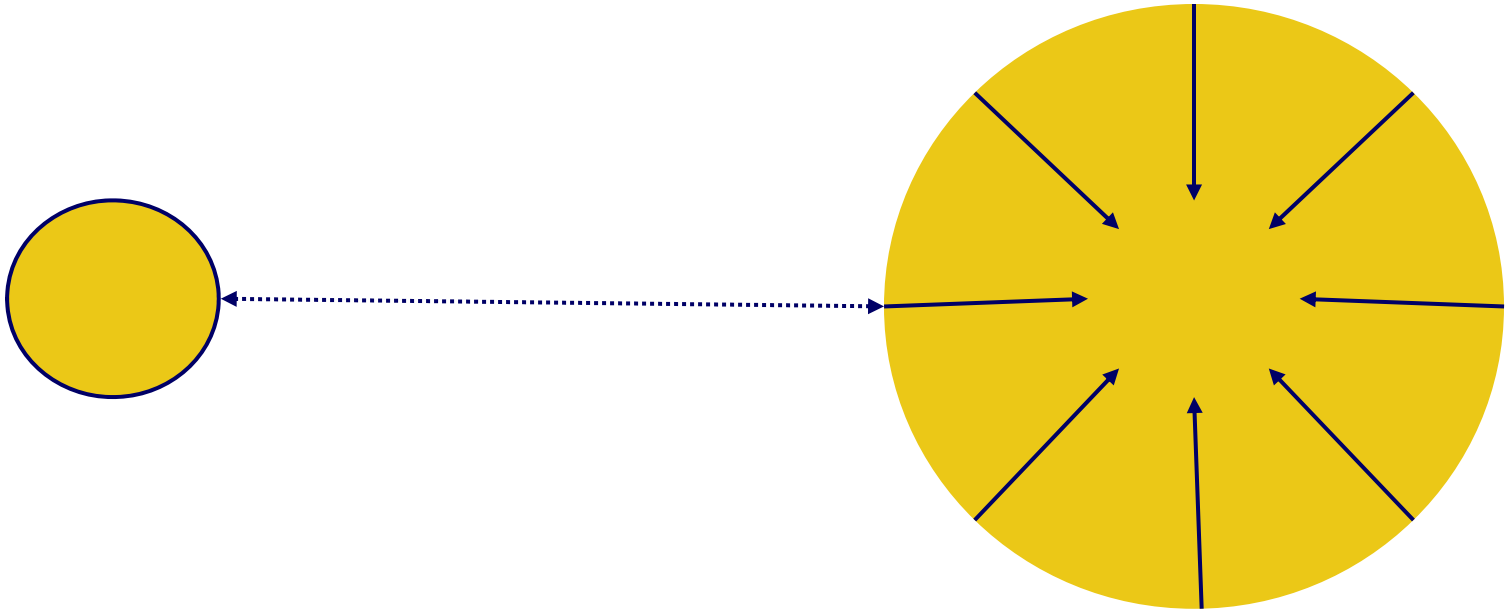
Knowledge sharing between sites



Knowledge sharing between sites

- **What causes these quality/productivity differences?**
  - Lack of communication?
  - Lack of knowledge sharing?
  - Relationship issues?
  - Relative group size?

Relative group size



Overall vision wrt knowledge sharing support: Feedback loop based on mining diverse data sources

advice

results, bugs, time, …

Summary

- Large variety of knowledge sharing issues
- Industry cooperation leads to challenging problems
- Software engineering becomes data-intensive