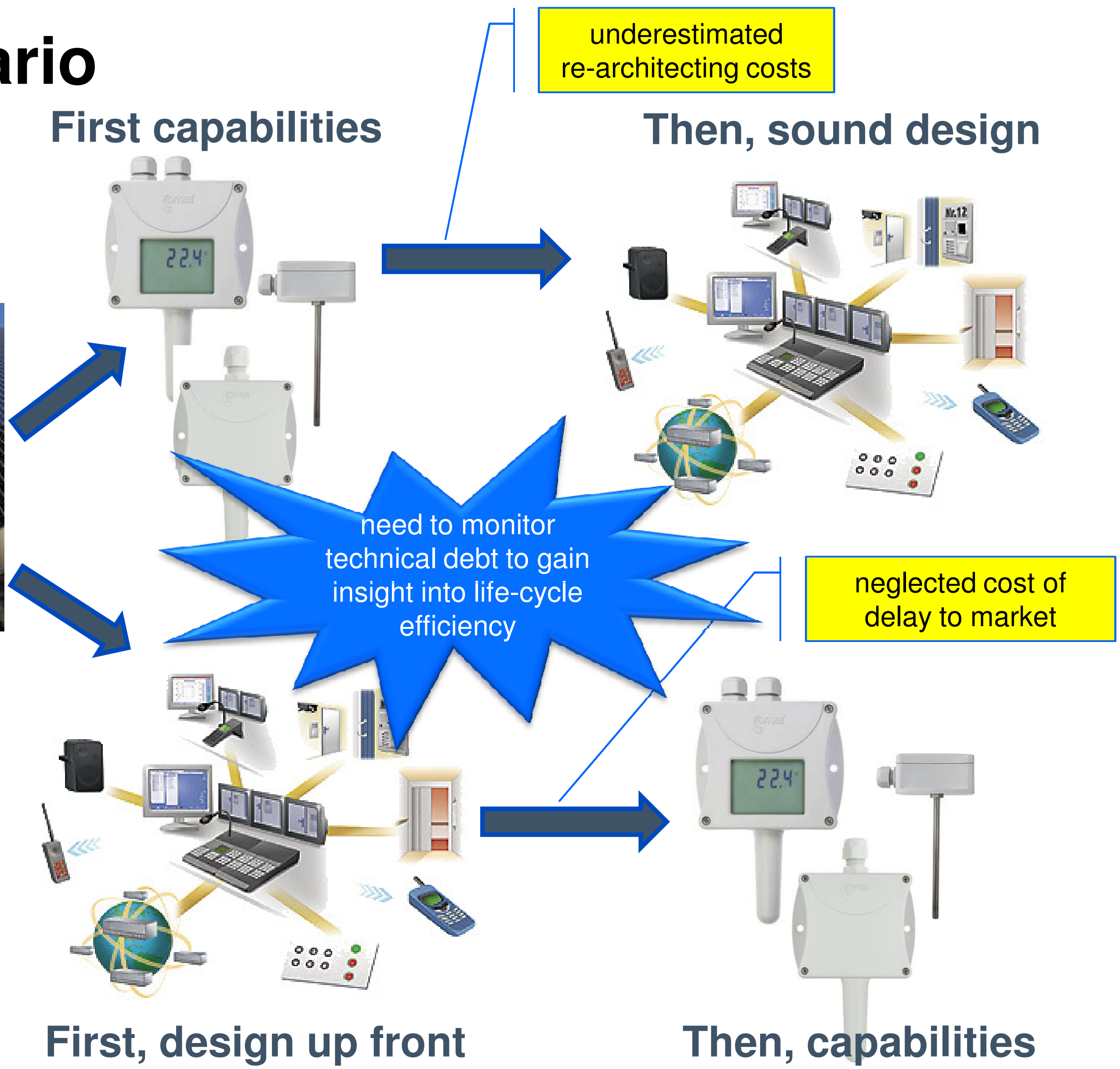
Quantifying the Value of Architecting within Agile Software Development via Technical Debt Analysis

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Nanette Brown, Robert L. Nord, Ipek Ozkaya (SEI)
Philippe Kruchten (University of British Columbia)

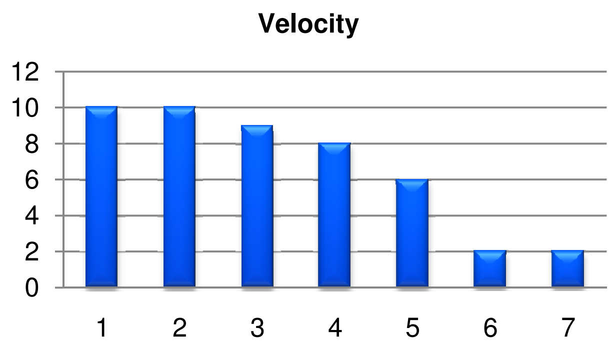
May 23, 2011

Guiding Scenario

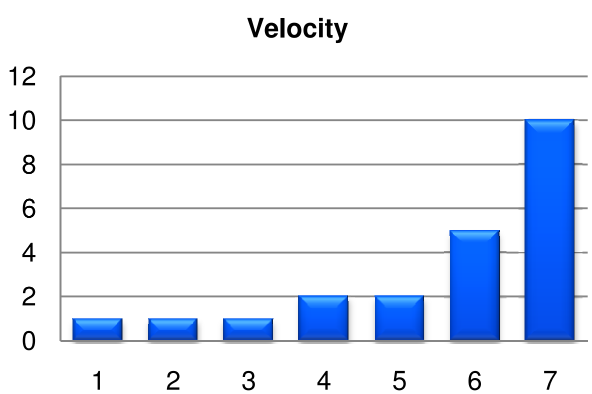


Future Directions – SEI Research

Focus on Value



Focus on Cost

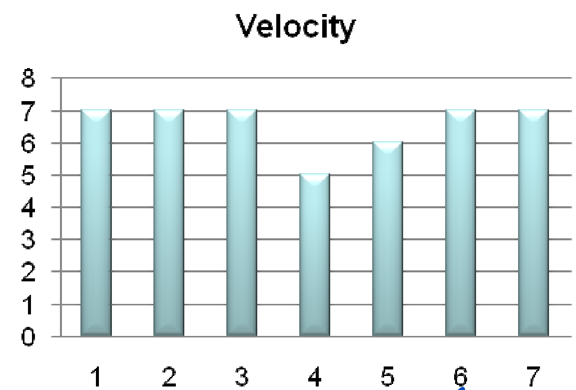


Manage architectural dependencies with dependency structure matrices

$root		1	2	3	4	5
03 Alarm Engine	1	20%				
06 Client Updates	2		20%			
07 Rule Processor	3	1		20%		
09 User Sessions Manager	4		1	1	20%	
14 Publish-Subscribe Bus	5	1		1	1	20%

Use propagation cost as a metric to monitor and focus development tasks

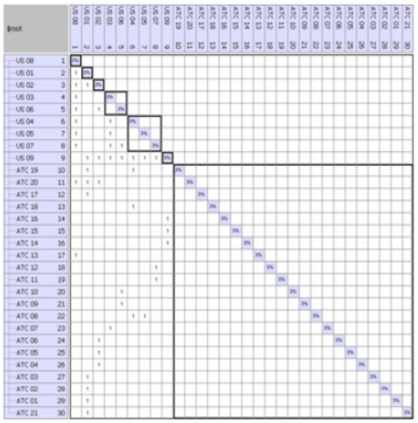
Focus on Integrated ROI



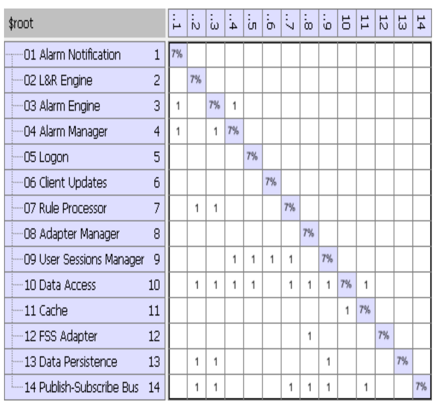
Ability to adjust course with empirical basis

Analysis and Management of Architectural Dependencies in Iterative Release Planning

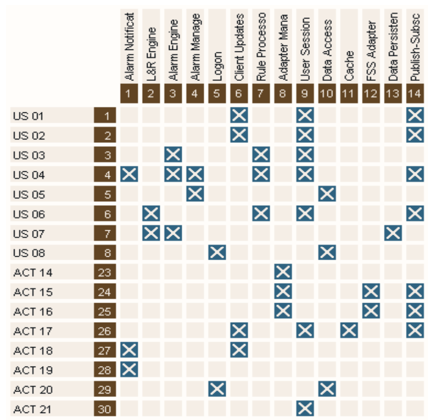
Modeling Dependencies



DSM - customer requirements

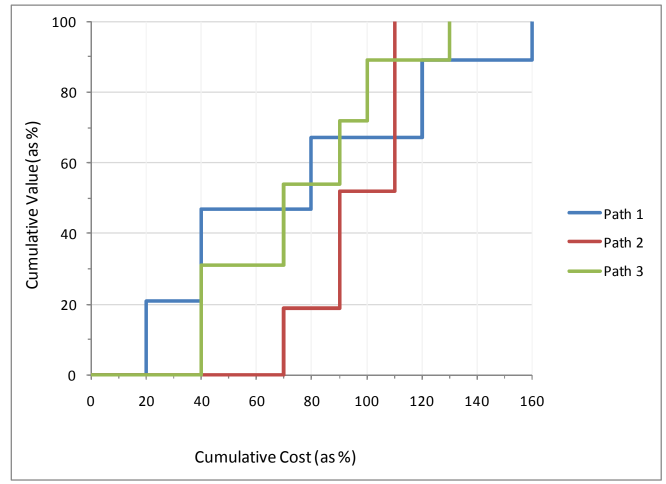


DSM – architectural elements



DMM – requirements to elements

Economic Models



Value of Capabilities Delivered over Total Effort

Metrics

$Tc_n = Ic_n + Rc_n$ — *Total cost as a function of implementation and rework cost*

$$Pc = \frac{\sum_{i=0}^{n} M^i}{n^2}$$

Propagation cost

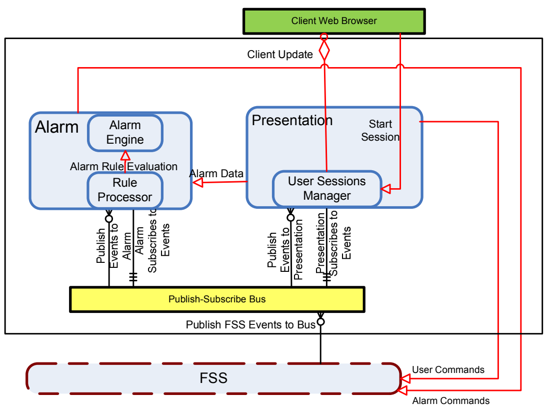
Technical Measures

Rework cost, Rc_n, for release *n* is computed as follows:

- Compute the rework cost associated with each new architectural element AE_k as the sum of the rework cost for each pre-existing AE_j

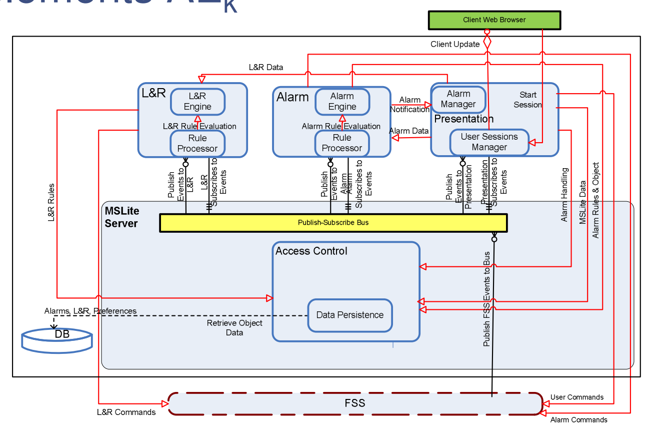
  $Rc(AE_j) = \#dependencies(AE_j\ ,\ AE_k) * Ic(AE_j) * Pc(n\text{-}1).$

- Sum the rework cost for all new architectural elements AE_k



$root		1	2	3	4	5
03 Alarm Engine	1	20%				
06 Client Updates	2		20%			
07 Rule Processor	3	1		20%		
09 User Sessions Manager	4		1	1	20%	
14 Publish-Subscribe Bus	5	1		1	1	20%

Release n-1



$root		1	2	3	4	5	6	7	8	9
01 Alarm Notification	1	11%								
02 L&R Engine	2		11%							
03 Alarm Engine	3	1		11%	1					
04 Alarm Manager	4	1		1	11%					
06 Client Updates	5					11%				
07 Rule Processor	6		1	1			11%			
09 User Sessions Manager	7				1	1	1	11%		
13 Data Persistence	8		1	1				1	11%	
14 Publish-Subscribe Bus	9		1	1			1	1		11%

Release n

Discussion Questions

The rework algorithm rework is directional in nature and represents an initial effort to formalize the impact of architectural dependencies.

- What are the appropriate proxies of complexity that affect the cost of change?

Rework cost is interpreted as a relative value, used to compare alternative paths and to provide insight into the improvement or degradation of architectural quality across releases within a given path.

- How do we incorporate uncertainty and the forecast of future rework in the model?

The ability to quantify degrading architecture quality and the potential for future rework cost during iterative release planning as each release is being planned is a key aspect of managing strategic technical debt.

- How do we characterize the economics of architectural violations across a long-term roadmap, rather than enforce compliance for each release?

Contact Information

RTSS Program
Linda Northrop
RTSS Program Director
Software Engineering Institute
Pittsburgh, PA 15213
lmn@sei.cmu.edu

Agile Architecting
Nanette Brown, Robert Nord, Ipek Ozkaya
Software Engineering Institute
nb, rn, ozkaya @sei.cmu.edu
Philippe Kruchten
University of British Columbia
pbk@ece.ubc.ca

Business Development
Austin Montgomery amontgom@sei.cmu.edu

SEI website at www.sei.cmu.edu/architecture

Additional Information

Nanette Brown, Robert Nord, Ipek Ozkaya, Manuel Pais. Analysis and Management of Architectural Dependencies in Iterative Release Planning. In: *Proceedings of the Working IEEE/IFIP Conference on Software Architecture* (WICSA) 2011.