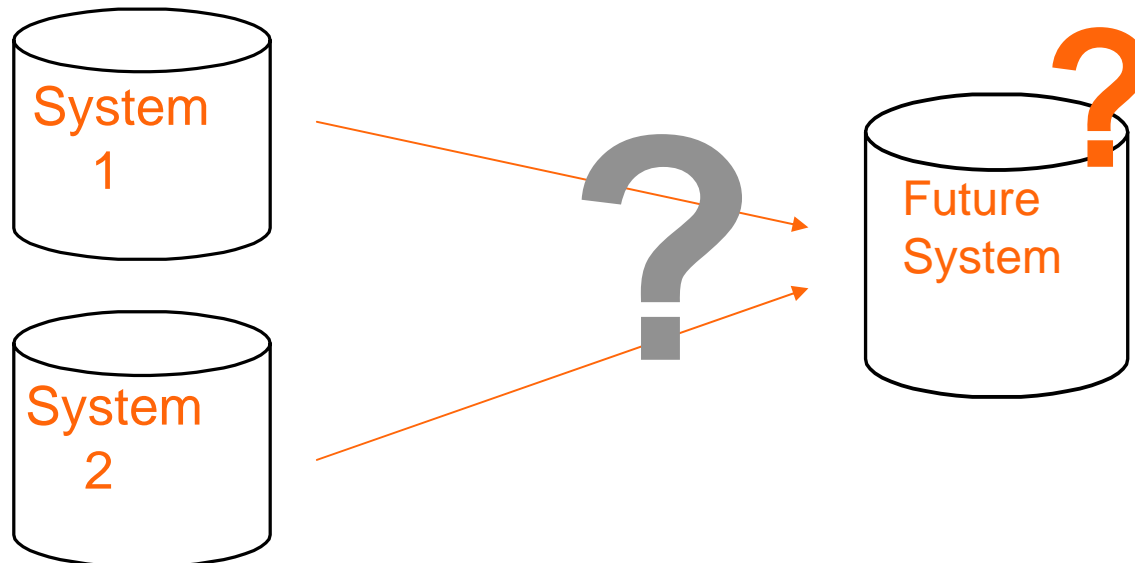


Software Systems In-House Integration Strategies: Merge or Retire – Experiences from Industry

Rikard Land, Laurens Blankers, Stig Larsson, Ivica Crnkovic
Mälardalen University
Department of Computer Science and Electronics

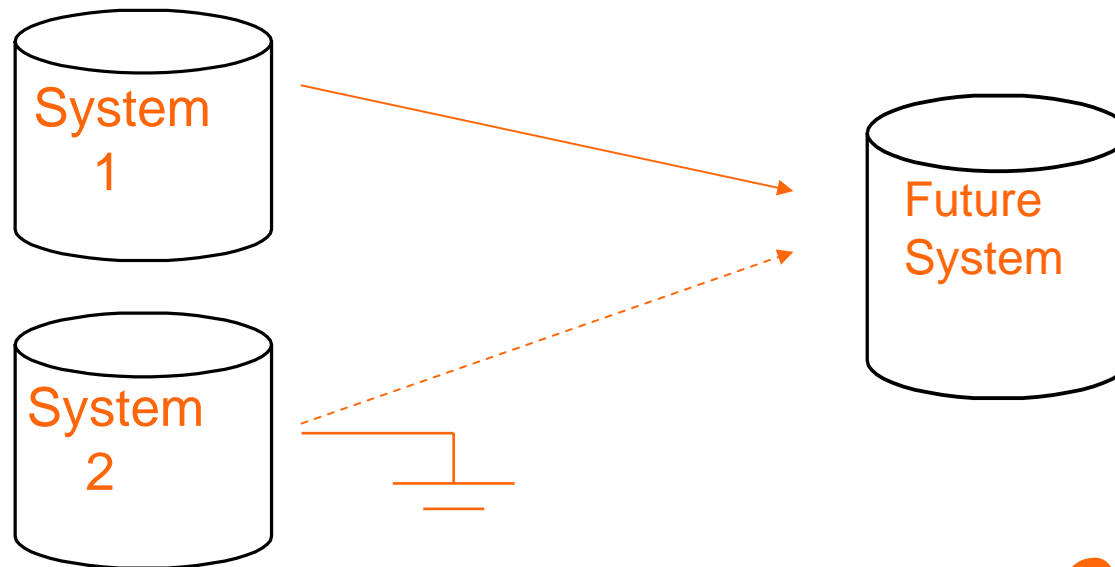
Background and Motivation

- Cause:
 - Different software systems developed in-house are growing
 - Organizations face new collaborations and mergers
- Effect:
 - In-house developed and controlled software systems address similar needs
 - Overlap in functionality



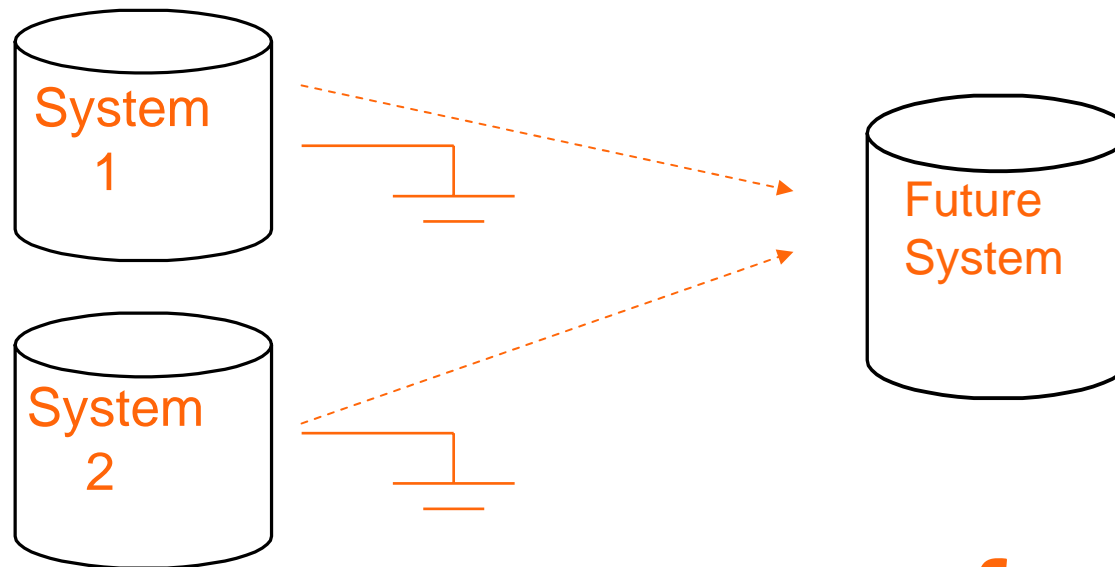
Oct 9 2005

4 strategies



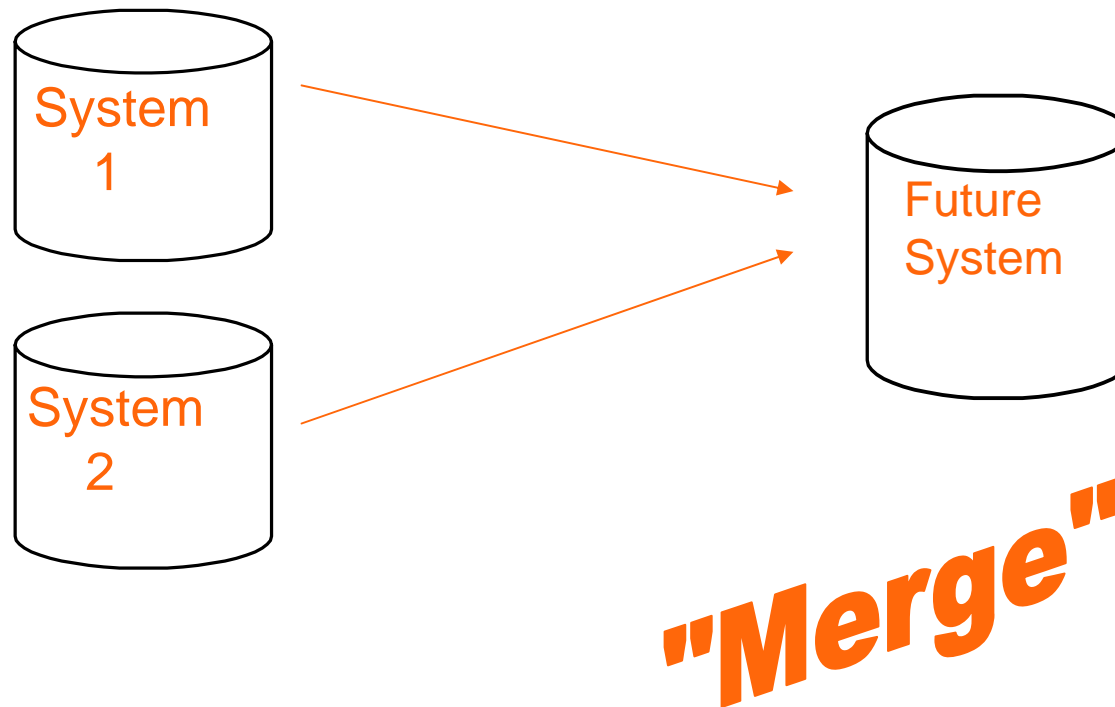
"Choose One"

4 strategies

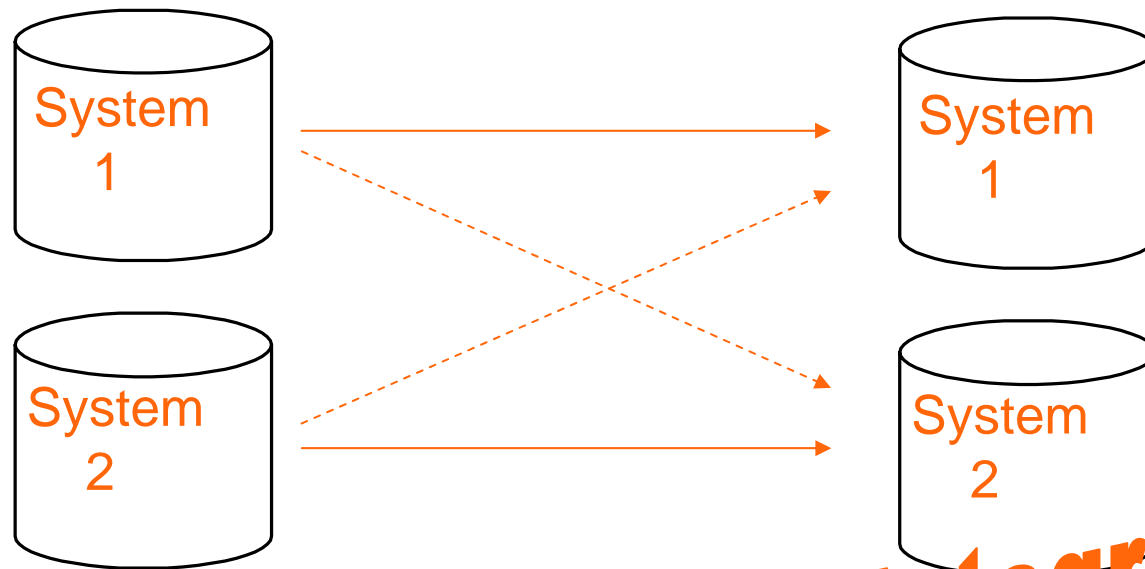


"Start from Scratch"

4 strategies



4 strategies



"No Integration"

	Organization	System Domain	Goal	Information Resources
A	Newly merged inter-national company	Safety-critical systems with embedded software	New HMI ^[1] platform to be used for many products	<i>Interview:</i> project leader for “next generation” development project (I _A)
B	Organization within large international enterprise	Administration of stock keeping	Rationalizing two systems within corporation with similar purpose	<i>Interview:</i> experienced manager and developer (I _B)
C	Newly merged international company	Safety-critical systems with embedded software	Rationalizing two core products into one	<i>Interviews:</i> leader for a small group evaluating integration alternatives (I _{Ca}); main architect of one of the systems (I _{Cb})
D	Newly merged international company	Off-line management of power distribution systems	Reusing HMI* for Data-Intensive Server	<i>Interviews:</i> architects/developers (I _{Da} , I _{Db}).
E1	Cooperation defense research institute and industry	Off-line physics simulation	Creating next generation simulation models from today’s	<i>Interview:</i> project leader and main interface developer (I _{E1}) <i>Document:</i> protocol from startup meeting (D _{E1})
E2	Different parts of Swedish defense	Off-line physics simulation	Possible rationalization of three simulation systems with similar purpose	<i>Interview:</i> project leader and developer (I _{E2}) <i>Documents:</i> evaluation of existing simulation systems (D _{E2a}); other documentation (D _{E2b} , D _{E2c} , D _{E2d} , D _{E2e} , D _{E2f})
F1	Newly merged international company	Managing off-line physics simulations	Possible rationalization by using one single system	<i>Participation:</i> 2002 (R.L.) (P _{F1a}); currently (R.L.) (P _{F1b}). <i>Interviews:</i> architects/developers (I _{F1a} , I _{F1b}); QA responsible (I _{F1c}) <i>Documentation:</i> research papers (D _{F1a}); project documentation (D _{F1b})
F2	Newly merged international company	Off-line physics simulation	Improving the current state at two sites	<i>Interviews:</i> software engineers (I _{F2a} , I _{F2b} , I _{F2f}); project manager (I _{F2c}); physics experts (I _{F2d} , I _{F2e})
F3	Newly merged international company	Software issue reporting	Possible rationalization by using one single system	<i>Interview:</i> project leader and main implementer (I _{F3}) <i>Documentation:</i> miscellaneous related (D _{F3a} , D _{F3b})

Exclusion Criteria

- Architectural Compatibility** *Static property of the systems*
 - The similarities (in some sense) of the existing systems influence how easy to combine parts
 - Architectural, because we need to discuss it at a high level
- Retireability (for lack of a better term)** *Negotiable*
 - Can any of the existing systems be retired?

		SFS	CO	EM	IM
Compatibility	High				
	Modest				
	None				
Retireability	All				
	Not all				
	None				

Exclusion Criteria 1: Architectural Compatibility

- Similar high-level structures seem to be a prerequisite for a *Merge*
- Similarity of “frameworks” (how components are defined)
- Differences in data model

Exclusion Criteria 1, cont: Architectural Compatibility

- Sources of similarities:
 - Common ancestry
 - Common (domain) standards Compatibility must be carefully evaluated and communicated prior to a decision
 - Not re-negotiable
 - Profound impact on the possible integration strategies
 - Counter-examples

Exclusion Criteria 2: Retireability

- Possible to retire any/some/all of the existing systems
→ *Choose One* or *Start from Scratch* possible
- Clearly, a matter of tradeoff
 - Problems/cost of retiring vs. benefits/savings
 - Problems/cost of not retiring vs. benefits/savings
- Question to be asked to various stakeholders: impact of retiring systems?

Strategy Exclusion in the Cases

	Retireability	Compatibility
A	All	None
B	Not all	None
C (initial)	None	Modest
C (final)	Not all	
D _{HMI}	Not all	None
*D _{Server}	(?)	<u>Modest (?)</u>
E1	All	Modest
E2	Not all	Modest
F1 (initial)	No	None
F1 (final)		
F2 _{Pre}	All	<u>None/Modest (?)</u>
*F2 _{2D}	All	Modest
F2 _{Post}	All	None
*F2 _{3D}	None	Modest
F3	All	<u>None/Modest (?)</u>

	SFS	CO	EM	IM
A	0			
B		0		
C (initial)				0
C (final)		0		
D _{HMI}		0		
*D _{Server}	(?)	(?)	0	(?)
E1	0			
E2		0		
F1 (initial)			0	
F1 (final)		0		
F2 _{Pre}	0		(?)	
*F2 _{2D}		0		
F2 _{Post}	0			
*F2 _{3D}			0	
F3	0		(?)	

External Validity?

- We have built our ideas (“hypothesis”) from the cases
 - We cannot use the same cases to validate the ideas (“hypothesis”)
 - Grounded theory [Strauss & Corbin]
- We can see three types of validation:
 - Validation 1: Multiple case study worth more than single [Yin]
 - Validation 2: Does it make sense? Is it useful? [Maxwell]
 - Validation 3: More cases. Not trivial to evaluate however.

Conclusions

- Four high-level strategies
 - No Integration
 - Start from Scratch
 - Choose One
 - Merge: Instant or Evolutionary
- Exclusion criteria:
 - Architectural compatibility
 - “Retireability” (focus questions to stakeholders in terms of retiring systems)
- Future work
 - Refine the notion of architectural (in)compatibility and how to succeed with merge