

Scaling Up Software Architecture Evaluation Processes



Liming Zhu, Mark Staples, Ross Jeffery

NICTA (National ICT Australia)

UNSW (University of New South Wales)

Liming.Zhu@nicta.com.au

May, 2008



Australian Government

Department of Broadband, Communications
and the Digital Economy

Australian Research Council

NICTA Members



Department of State and
Regional Development



The University of Sydney



Queensland
Government



Griffith
UNIVERSITY



Queensland University of Technology



THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA

NICTA Partners

- Background and Motivation
 - What is scaling up processes?
 - Ultra-large-scale system challenges
 - Reference architecture for Australian Lending Industry
- Evaluation Process for Rule-centric Architecture (EPRA)
 - Introduce scalability into processes
 - process component customization
 - loosely-coupled process components and process interactions
- Case Study
 - Evaluate the reference architecture using EPRA
- Discussion
- Conclusion

Background – Scaling Up Processes



- Resource size (development team)
 - Communication and training overhead
 - Certain work is difficult to be performed by more than 1 person
 - Solution:
 - Scale-up: improve individual's capabilities; free up individual's workload
 - Scale-out: add more people
 - Challenges: software modularity, the nature of the work and overhead
 - Example studies: scalability of agile methods [1]
- Software size (architecture & code) – ultra-large-scale systems
 - Software engineering techniques
 - **Development processes**

[1] L. Zhu and M. Staples, "Situational Method Quality," in IFIP WG8.1 Working Conference on Situational Method Engineering: Fundamentals and Experiences (ME07), 2007.

Motivation



- Australia Lending Industry XML Initiative (LIXI)
 - provide guidance and development processes for the lending industry
- System Characteristics – Ultra-Large-Scale Systems (ULSS)
 - Decentralization
 - both systems and development processes
 - Inherently conflicting requirements
 - complexity, data sharing and intermediaries
 - Continuous evolution with heterogeneous elements
 - embedded/enterprise/people
 - No clear people/system boundary
 - different levels of technology sophistication
 - Horizontal interaction with other industries

- Data standards in XML schemas (Domain and XML experts)
- Business process solutions (NICTA)
 - New business process languages and approaches [1][2]
- Architecture solutions: rule-centric architecture (NICTA)
 - Rule-centric: instantiate design tactics; map to multiple structural arch.
 - Social-technical: use economic incentives, foster diversity and diffuse decision making power
- **Software process solutions (NICTA)**
 - **Evaluation processes for Rule-centric Architecture (EPRA)**
 - Development processes for ultra-large-scale systems (undergoing)

[1] L. Zhu, L. Osterweil, M. Staples, U. Kannengiesser, and B. I. Simidchieva, "Desiderata for Languages to be Used in the Definition of Reference Business Processes," *International Journal of Software and Informatics*, vol. 1, pp. 37-66, 2008

[2] X. Xu, L. Zhu, Y. Liu, and M. Staples, "Resource-Oriented Business Process Modeling for Ultra-Large-Scale Systems," in *Second International Workshop on Ultra-Large-Scale Software-Intensive Systems (ULSSIS 2008)*, co-located with ICSE,

- Architecture evaluation processes/methods
 - Examples: ATAM/SAAM, ALMA ...
 - Suitable to well-defined structural architectures
 - Generic processes and process components
- Process tailoring and method engineering
 - Limited criteria for tailoring: project characteristics
 - Resource size (not software size)
 - Domains and quality requirements (not about scaling up)
 - Limited capabilities in achieving process variability
 - Method engineering: merge, bridge, replace and extend process components (method chunks)
 - Mostly functional (not non-functional – scalability)
 - Workflow modelling: control flows do not scale

Starting Point: Ten Process Components



- *Explicit elicitation of business goals.*
- *Active stakeholder participation and communication.*
- *Explicit elicitation of architecture documentation and rationale.*
- *Use of quality attribute scenarios.*
- *Mapping of quality attribute scenario onto the architecture representation.*
- *Representation of design primitives, called tactics.*
- *Use of templates to capture information*
- *Explicit elicitation of costs and benefits*
- *Use of quality attribute models for architecture analysis.*
- *Use of quality attribute design principles derived from the quality attribute model.*

Introducing Scalability into Process Components and Processes



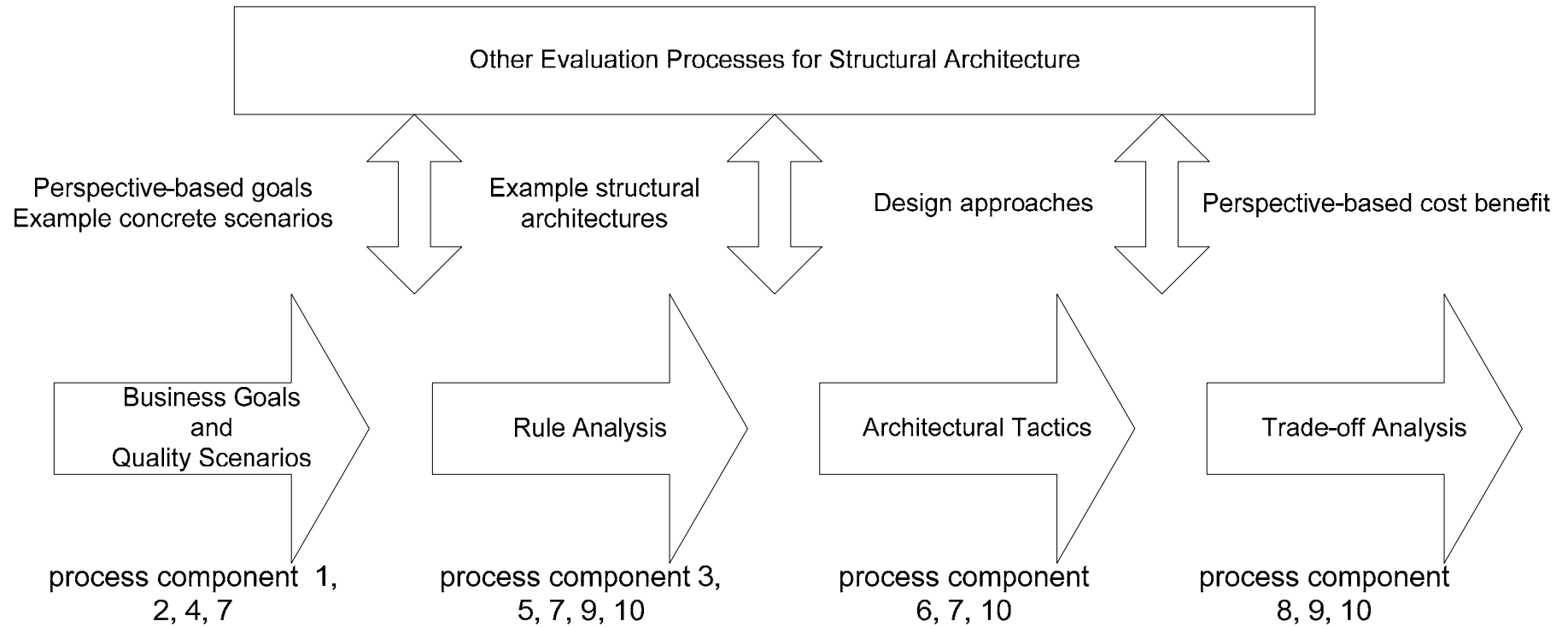
- **Abstraction**
 - Switch between diff. levels of abstraction: rule vs. structure
- **Contextualization – process parameterization**
 - Perspective-based: conflicting stakeholder views
 - Technology-based: different technology bindings
- **Loose coupling (among process components)**
 - Under-specify control flow
 - Share contexts
- **Process interaction**
 - Interaction between rule-based and structure-based evaluation processes

Some Examples



- Explicit elicitation of business goals - *Contextualization*
 - Perspective-based goals
- Active stakeholders participation - *Abstraction*
 - Rule-level understanding and structure-level understanding
- Phase-based evaluation processes – *Loose coupling*
 - Not step-wise; weak dependency
- Artefact flows between processes – *Process interaction*
 - rule evaluation <-> exemplary structure evaluation

Evaluation Process for Rule-centric Architecture (EPRA)



Case Study – Evaluating LIXI Reference Architecture



- Phase I: Business Goals and Quality Scenarios
 - Reduce development cost - Manage flexibility
 - Reduce development cost - Distributed development processes
- Phase II: Rule Analysis
 - Semantic alignment rules
 - Minimum service interface rules
- Phase III: Architectural Tactics
 - Extracting tactics from rules
- Phase IV: Trade-off Analysis
 - Different quality attributes
 - Different perspectives
- Outcome
 - Improvement over the rules
 - Smooth process execution among different participants

Discussion



- Reusing process components is very effective.
 - Need more sophisticated process customization
- Evaluating reference architectures for ultra-large-scale systems is very explorative.
 - Need loose coupling among process components
- Large software size does affect process scalability.
 - Need process abstraction
 - Need new techniques for ultra-large-scale systems



Conclusion



- Ultra-Large-Scale Systems (ULSS) is a reality.
- Different scaling up challenges: resource size and software size
- Different ways of scaling up
 - “Software processes are processes too” -> Techniques for achieving software scalability may apply to processes too:
 - Process contextualization and parameterization
 - Process abstraction
 - Loose coupling
 - Process interaction
 - Software modularity
- Scaling up software architecture evaluation processes
 - First attempt to scale up
 - Feasible and effective

An Analogy



	Social Networking	Lending Industry
Players	MySpace, Facebook, LinkedIn, Plaxo, 	Lenders, Brokers, Insurers, Aggregators...
Federated business-technical model	Google OpenSocial 	LIXI Business Processes LIXI Reference Architectures
Functions	Distribute people info + Pub/Sub + Persistence	Distribute loan info + perform complex lending processes

New software engineering processes for ultra-large-scale systems needed