ECSA 2010

Fourth European Conference on Software Architecture

IT University of Copenhagen, Denmark

23.-26.08.2010

Content

- Introduction

- Some things relevant to us
  - Keynotes
  - Some Papers
  - 8th NW-MDE

- Conclusions

Introduction

- A series of European Workshops on Software Architecture started from 2004 (UK)
  - Changed into Conference format from 2007 (Madrid)
- Content
  - from 135 abstracts and 106 papers submitted
  - 19 full research and 31 other shorter papers were accepted
- Keynotes
  - Jan Bosch & Philippe Kruchten & Jim Webber
- This time joint event with 8th Nordic Workshop on Model Driven Software Engineering

Keynotes – Jan Bosch

- speed is the foundation of everything else (efficiency is the by-product of speed)
  - small (max. 3 persons) independent self-directed teams
    - cost of overlapping work is less than cost of synchronization/planning
  - continuous deployment – no release cycles
    - anything anytime should be possible to add, not breaking the system
  - software ecosystem – open platforms
- architecture must
  - focus on simplicity (hide, platformize, automatisize)
    - no options for developers!
    - no versions of components – single version of everything!
  - provide compositionality (move from integration to composition)
  - minimize dependencies (decoupling of components & teams)
  - ensure end-to-end quality and fight design erosion
- delightful products

Keynotes – Philippe Kruchten

- Knowledge is an asset
  - Bacon: “Knowledge is power”
- Knowledge evaporates
  - “intellectual capital has legs and it walks ...”
  - use documentation, process and tools to fight knowledge loss
- Often architectural knowledge is tacit
- Knowledge management strategies
  - codification (repository) – document and make available
    - select only essential, for which there is real reader
  - personalization (people) – know who knows!
    - use incentives to make people sharing information

Keynotes – Jim Webber

- usual agile FUD toward enterprise-level tools, but some good advices
- don't absolve yourself from thinking!
  - no to gut feeling – guts are for “food processing”, use your head!
- model and analyze before deciding
  - build working models (they call these “spikes”)
    - no alternatives were evaluated
    - there is no need for queuing = with enough iron it worked without it
  - measure continuously the limiting quality characteristic(s)
- always present large numbers!
  - ~1 000 000 000 req/month = ~390 req/sec
    (100 ÷ 1 000 TPS is considered medium)

Some Papers

- Customer Value in Architecture Decision Making
  - What
    - customer value links architecture decisions directly to the business goals
    - customer value quantifies the value of architecture change in money
  - How to use
    - model customer business (in different segments) and
    - analyze impact of architecture changes (quality improvements) on customer business
    - make time-dependent architecture change scenarios and
    - analyze impact of architecture change scenarios on customer business
- Impact Evaluation for Quality-Oriented Architecture Decisions regarding Evolvability
  - plug-in and pipes&filters correspond mostly to evolvability
  - blackboard (central DB!) has negative effect toward evolvability

Some Papers

- Linking Design Decisions to Design Models in MBSD
  - Eclipse + QEDwiki (for architectural decisions)
- Lightweight and Continuous Architecture Software Quality Assurance using the aSQA Technique
  - SEI quality attributes
    - availability, perfromance, modifiability, testability, security, usability
  - for each estimate (from 1-5): target, current, importance
  - then calculate
    - health = 5 – max(0, target – current)
    - focus = [(6 – health) x importance/5]
- ... several ...
  - run-time availability of system (architecture) description
  - support for on-line replacement of components
- Independently Extensible Contexts
  - support for unanticipated extensions
  - network of objects describing the domain (context)

Some Papers

- Explaining Architectural Choices to Non-Architects
  - use radial diagrams, for quality attributes, for comparing different alternatives
  - let them decide (or let them believe that they decide)
- Experiences in Making Architectural Decisions during ...
  - keep architectural description separate from requirements
  - do decisions at right time (as risk management)
  - decision classes:
    - in form of structure
    - in form of technique (how to make structures)
    - in form of (essential) requirements – postpone/delegate decisions
    - in form of process prescription (what to do)
  - decisions need information
    - always model and try things out before the decision
  - link decisions to requirements

Some Papers

- Unifying Software Architecture with its Implementations
  - design is lost in the code – programming languages don't have any constructs that correspond to design structures
  - use code to describe the design
    - naming conventions
    - build systems (system models – maven)
    - component models (Spring, OSGi, ...)
    - metadata (Java annotations, Doxygen, ...)
    - comments
    - AOP
  - architecture recovery is doomed, because the knowledge is not there – it has been thrown away

8th NW-MDSE Keynote

- Language abstractions are insufficient to express what platforms offer
- Great increase in computing power, but software is still made as 40 years ago
- Biggest driver of architecture is Conway's Law
  - usually common domain expertise group is missing!
- Model-Driven Softare Development = Speed
  - avoiding duplicate code
  - hiding platform complexity
  - reusing expertise
- When gaining power/speed we loose control
- Model-Driven Techniques create pressure to specify/define requirements

8th NW-MDSE

- Study of model usage (in car industry)
  - process document-centric (required deliverables are documents)
  - requirements are perceived as most important, but
  - models require largest effort (61% vs 23%)
    - same in pure MDD project (59%) than in non-MDD projects (61%)!
  - models are not systematically reused (an one-time effort)
- Repository support for free form tools
  - often company official modeling tools are not used
    - many informal tools and notations are used instead
    - information will be lost after project
  - MS Office is highly available/accessible and often used as informal tool for modeling – model repository support is added
- Clone detection in Domain Models
  - as models get bigger, problems will be different

Conclusions

- Speed – basis for everything else
  - seek speed – efficiency follows
- Simplicity – fight with complexity everywhere
  - hide, automatisize, remove options
  - fight architecture erosion
- Decouple components, teams and organizations
  - continuous independent deployment
- Manage Architectural Knowledge
  - Personalize – communities of practice (incentives to share)
  - Codify – document the essential/critical
- Ensure end-to-end quality – automatisize QA

Thank You!

Questions?

Conclusions from last time!

- Not Documenting, but Understanding
- Design is for Humans
- Problem Structure should define Solution Structure
- Ideal architect is objective
  - he doesn't have any favorite techniques, and
  - all he does, has rationale