Liveware

SATURN 2014
Portland, Oregon | May 5–9, 2014

Combining Architectural Methods for Building a Reference Architecture for Ground Radar Monitoring Systems

Methods & Tools: Experience Report

Alejandro Bianchi, Liveware IS, Argentina
Andrés Diaz-Pace, UNICEN University, Argentina
Leonardo Seminara, Liveware IS, Argentina

Outline

- Context
  - Domain
  - Existing problems & challenges

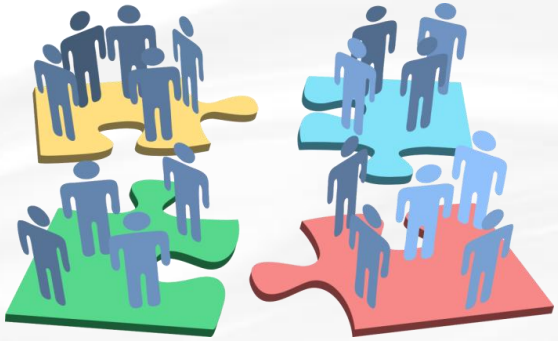
- **Proposed architectural strategy**
  - Reference architecture
  - Business goals + QAW + ADD + Architectural Evaluation
  - Tailoring of methods
- Lessons learned
- Conclusions

***Disclaimer**: Due to confidentiality agreements with the customer, we cannot disclose technical details of the system/organization. Information and examples have been anonymized or even changed in parts of this presentation.*

Ground radars & monitoring (GRM)

- Standard communication protocols
  - periodically receives data from radar
- **Main purpose**
  - Process events from the radar → **telemetry**
  - Analyze telemetry, trigger alarms
- Users
  - **Operator** & System maintainer
  - The operator trusts the SW as if it were (a mirror of) the HW (radar)
- **Main Qualities**
  - Fidelity, performance, "diagnosticability" of problems

The organization



- **Multi-stakeholder landscape**
  - Physicists, nuclear engineers
  - System engineers, electronic engineers

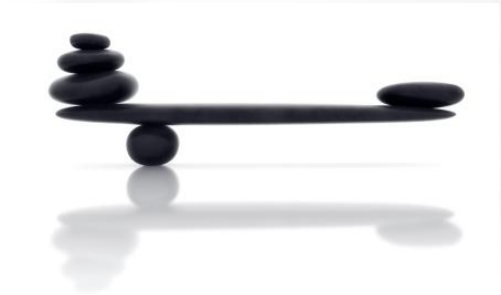
  + A RELATIVELY NEW/SMALL SOFTWARE DIVISION

- An existing GRM system (brownfield), already in operation
  - 2nd version (developed by a contractor)
  - Little design documentation
  - Maintenance & evolution problems

- Bottom line: Software can add value to the organization
  - **but, this was not always perceived in day-to-day activities**

Why an architecture-centric solution?

- Different types of radars/products → **software family**
  - Technology and deployment variations
  - Kind of information to be collected/analyzed



- Organizational & development factors
  - Speed up development cycles **without sacrificing quality**
  - Need to engage stakeholders/contractors for (future) products
  - Limited (and overwhelmed) software development personnel

→ **external (architecture) team: Liveware IS**

→ **architectural principles as a leverage for product evolution**

Proposal: a Reference Architecture

- It enables a **systematic reuse** of domain knowledge and components when developing concrete architectures and systems

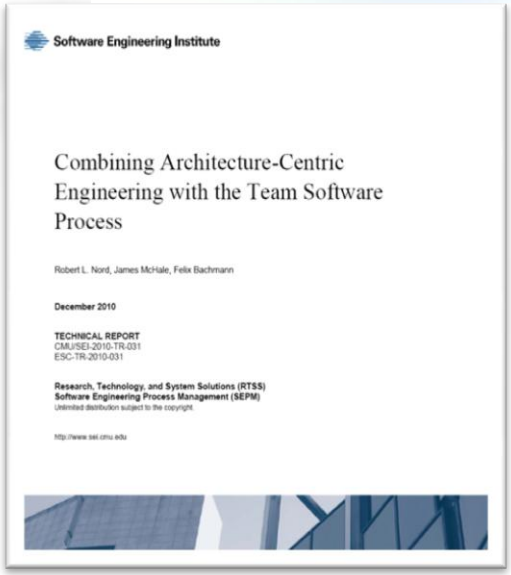
- **Informative role** (knowledge sharing)
  - Plus some design prescriptions

- Include (some) practices of Software Product Lines
  - Take advantage of existing system → **reusable assets**

Our technical approach

1. Identify business goals & relevant quality attributes
   - **QAW** (extended with **business goals + risk assesment**)

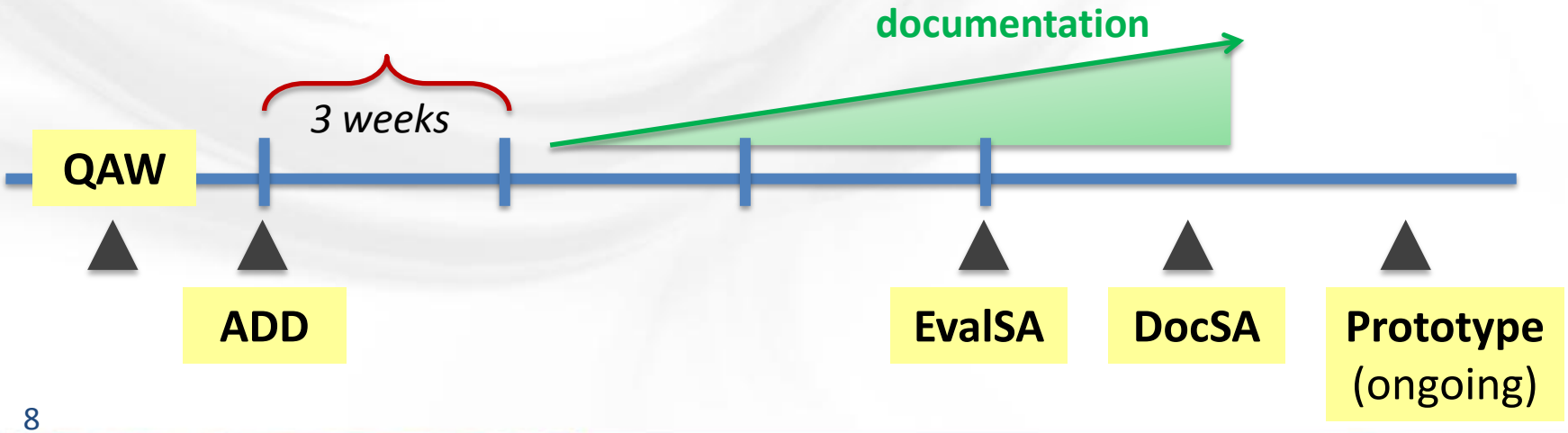
2. Create the reference architecture
   - Mine assets from existing application code
   - **ADD** (with support from EA tool)
   - Produce architecture documentation (kind of **Views & Beyond** + variation points)



3. Evaluate the architecture
   - **ATAM-oriented**
   - Provide technology guidelines

People & process considerations

- Architecture team
  - On our side: 3 people for **QAW → ADD → Evaluation**
  - On the customer side: Technical liaison + small group of developers/maintainers, who worked remotely
- Iterative & incremental strategy (~5 months)
  - 3-week iterations (with status-sync meetings every 2 weeks)
  - Plan, measure and report “design work”



1. Analysis of quality attributes

- Identification of business goals (BGs) for the organization
- Trace those BGs to quality attributes
- Generate & prioritize quality-attribute scenarios

- Identify **architectural plan**
  - Discussion of as-is and to-be states for GRM system
  - Layered architectural pattern
  - Extension: identify **risks** of the **architectural plan**

1. Analysis of quality attributes (cont.)

- **Challenges** for QAW
  - Involving all stakeholders was difficult, but ultimately fruitful
  - They were able to agree on **priorities for the quality scenarios**

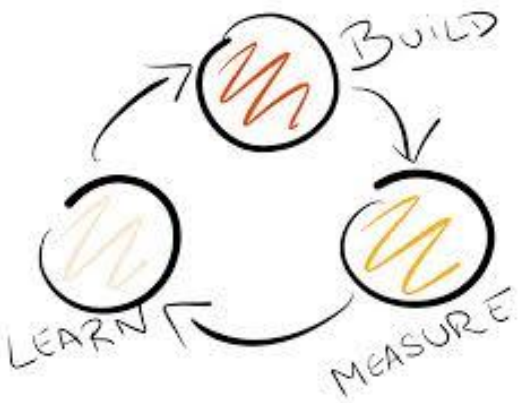
  (more on this later)

- **Technical problems** detected in architectural plan
  - Performance issues associated to the telemetry processing pipeline
  - High coupling between presentation and business logic
  - User-based access policies (operator, maintainer)

- These risks/problems were useful to inform architectural design phase

2. Design of the architecture

- 3 ADD iterations
  - 4 scenarios per iteration
  - High priority first
  - One architect had prior experience with SCADA systems



- A **review/feedback meeting** with technical liaison after each iteration

- Scenarios as “evidence” for
  - measuring progress
  - conducting periodic reviews with technical liaison

Driving the ADD process with EA

- **C&C views and sequence diagrams** were elaborated first
  - runtime architectural drivers
  - later, mapped to module and deployment views
- EA support
  - Profile with module, C&C and deployment viewtypes
  - Guidelines for a (limited) UML, organized around scenarios
- Design decisions were explicitly captured

- Challenges
  - Functional specifications were insufficient to define **interfaces and data model → important for reference architecture**

3. Architecture evaluation

- All stakeholders present (even “new” stakeholders)
- Walkthrough for each key scenario
  - Architectural team explains design rationale and presents supporting diagrams
  - Discussion of pros/cons of decisions made
  - Issues capture: sensitivity points, tradeoff points, risks, no-risks,
- **Metaphor: mappings drivers on architectural plan**

- New functionality related to **business goals** came out
  - Some could be accommodated by the actual design solution
  - Some other was out of scope, other needed a major system re-structuring

Impact on development strategy

- How to implement the new architecture?
  - Levels of reuse of existing assets
  - Role of evolutionary development

- Technological decisions
  - Broker
  - DB options
  - Recommendations: Comparative table with pros/cons of technologies was presented to the stakeholders for discussion

Lessons learned - pros

- **Convergence of stakeholders**
  - Discussion/articulation of different viewpoints about GRM products

  "We should do this (workshop) for other products as well!"
- Scenarios as "actionable items" in the lifecycle
  - Linkage among architectural methods
  - Measure of progress and risk indicators
  - Make the (design) work visible to customers
- Reinforce the role of Software Development Division
  - Also supported by Software Engineering practices
- **RSA as a proxy for a long-term vision of products**
  - Not just *"a re-structuring of something that works already"*

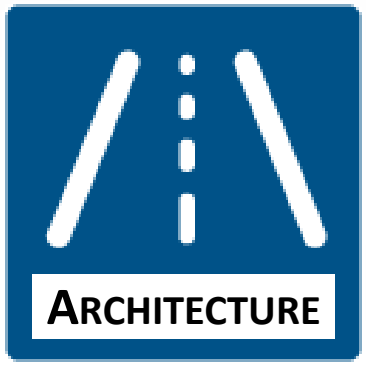
Lessons learned - cons

- **Little functional information** can be a bottleneck
  - Data model for RSA must still be defined
- **Architecture reconstruction from legacy not as expected**
  - It can still inform the current implementation of some components (e.g., communication drivers)

- How to do “evolutionary development” without compromising the architecture?
  - Room for conformance techniques
- How much architecture documentation is enough?
  - Wiki approach, consideration of stakeholders’ information needs

Follow-up & Conclusions

- ATAM-like analysis to guide prototyping activities
  - Currently developing an end-to-end prototype derived from one of the key quality drivers
- Importance of **visual metaphors**
- Product development strategy is being re-considered
  - Architecture-centric vision is a key part
  - New markets envisioned (e.g., medical devices)



- Need to have a **measurement framework for architecting activities**

Thank you!

alejandro.bianchi@liveware.com.ar

adiaz@exa.unicen.edu.ar

leonardo.seminara@liveware.com.ar