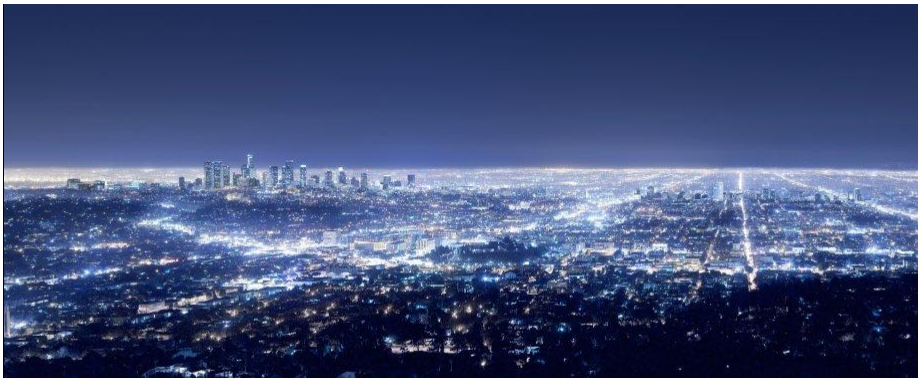


Olaf Zimmermann, ABB Corporate Research, Switzerland
With contributions from Heiko Koziolek and Martin Naedele

Making Architectural Knowledge Sustainable

Industrial Practice Report and Outlook

Power and productivity for a better world™ ABB

Abstract

Industrial software solutions such as Supervisory Control and Data Acquisition (SCADA) systems for power grids are complex systems with advanced quality requirements. Domain-specific design challenges include multi-decade life cycles of managed devices, advanced security regulations, and real-time requirements. Moreover, domain-specific software solutions have to be integrated with general-purpose ones, e.g., asset management packages and web portals.

ABB Corporate Research supports product development units in applying novel software technologies and software engineering methods effectively, striving for products that are attractive to customers and efficient to operate; this presentation shares lessons learned from such initiatives. For instance, we introduce a software-sustainability guide that profiles and packages a set of proven practices, including requirements elicitation with multi-level use cases and quality attribute scenarios, lightweight approaches to architecture documentation and evaluation, and state-of-the-art testing methods. Finally, we discuss how architectural knowledge management, e.g., sharing design-decision rationale, can be combined with existing practices to further improve project collaboration.

ABB

Agenda

- Context
  - Software and software research at ABB
  - Distributed process control systems
  - Domain-specific design challenges
- Sample projects and initiatives
  - Q-ImPrESS performance modeling
  - Software development improvement initiatives
- Architectural Knowledge Management (AKM) practices
  - Decision rationale – past and present
  - Capturing advice – relevant issues, good justifications
  - Towards an sustainability guide for AKM
- Summary and roadmap

ABB

Agenda

- **Context**
  - Software and software research at ABB
  - Distributed process control systems
  - Domain-specific design challenges
- Sample projects and initiatives
  - Q-ImPrESS performance modeling
  - Software development improvement initiatives
- Architectural Knowledge Management (AKM) practices
  - Decision rationale – past and present
  - Capturing advice – relevant issues, good justifications
  - Towards an sustainability guide for AKM
- Summary and roadmap

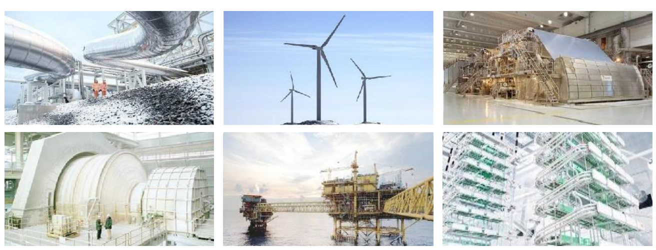
ABB

A Global Leader in Power and Automation Technologies

- 133,600 employees in about 100 countries
- $37.990 million in revenue (2011)
- Formed in 1988 merger of Swiss and Swedish engineering companies
- Predecessors founded in 1883 and 1891
- Publicly owned company with head office in Switzerland

ABB

Power and Productivity for a Better World
ABB's Vision



As one of the world's leading engineering companies, we help our customers to use electrical power efficiently, to increase industrial productivity and to lower environmental impact in a sustainable way.

ABB

How ABB is Organized
Five Global Divisions

Power Products	Power Systems	Discrete Automation and Motion	Low Voltage Products	Process Automation
$11.2 billion	**$6.5 billion**	**$5.4 billion**	**$4.1 billion**	**$7.8 billion**

2009 revenues (US$; pro-forma figures for automation divisions)

- ABB's portfolio covers:
  - Electricals, automation, controls and instrumentation for power generation and industrial processes
  - Power transmission
  - Distribution solutions
  - Low-voltage products
  - Robots and robot systems

Software at ABB
Software – Intelligence for ABB Products

Software is part of most ABB products – from the very small to the very big

Pressure sensor

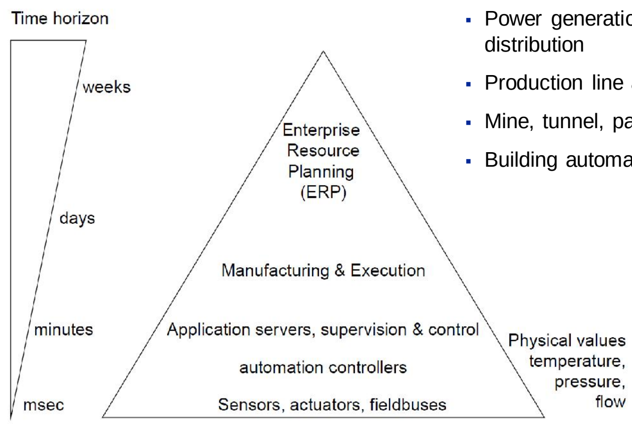
Industrial control system

Power grid control center

… and all have highest requirements for

- Real-time performance
- Reliability
- Long lifetime
- Remote connectivity

Collaborative Process Automation Systems
Automation Pyramid

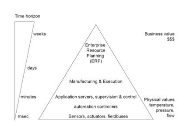


- **Examples**
  - Power generation, transmission, and distribution
  - Production line at car manufacturer
  - Mine, tunnel, paper mill
  - Building automation

Source: M. Hollender, Collaborative Process Automation Systems, ISA 2010

ABB

Collaborative Process Automation Systems
Automation Levels

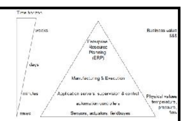


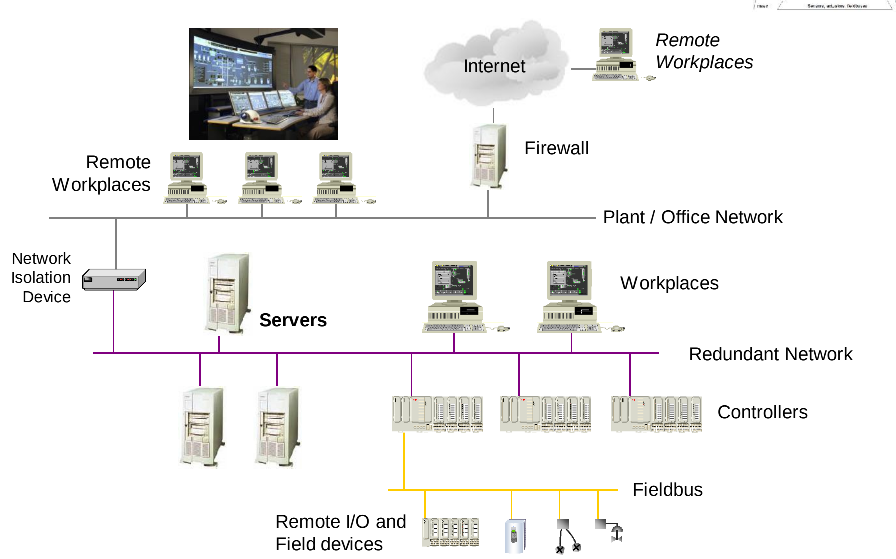
Level	Hardware/Software Systems	Typical Responsibilities
Enterprise Resource Planning (ERP)	Enterprise Resource Planning (ERP), Enterprise Asset Management (EAM)	Production planning (coarse), order management logistics, plant production and scheduling, asset management
Manufacturing & Execution	MES, MIS, LIMS	Production planning (detailed), production data and gathering, KPIs, materials management, quality mgmt Scheduling, reliability assurance
Application servers, supervision & control	Distributed Control System (DCS), Process Control System (PCS) Human Machine Interface (HMI), Supervisory Control and Data Acquisition (SCADA)	Operate and observe, recipe management, Archiving of measurement data (historian)
Automation controllers	SPS, control loops	Batch control, continous control, discrete control
Sensors, actuators, field buses (and managed process)	Process signals, I/O modules, fieldbuses Parallel wiring or intelligent systems like: AS-Interface	Interface to technical production process via signals Simple and rapid data collection, moslty binary signals

ABB

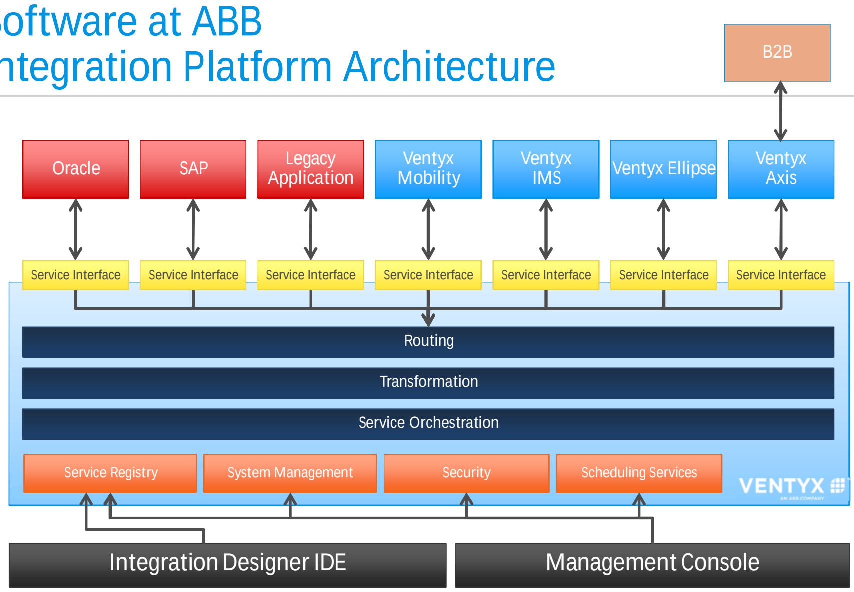
Industrial Automation: Process Control Systems (PCS)

PCS Infrastructure (Operational Model)

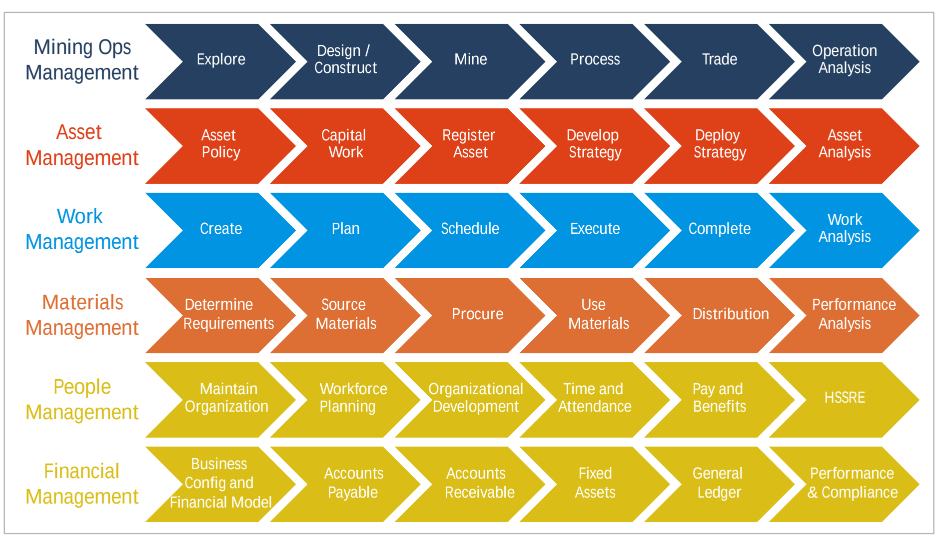




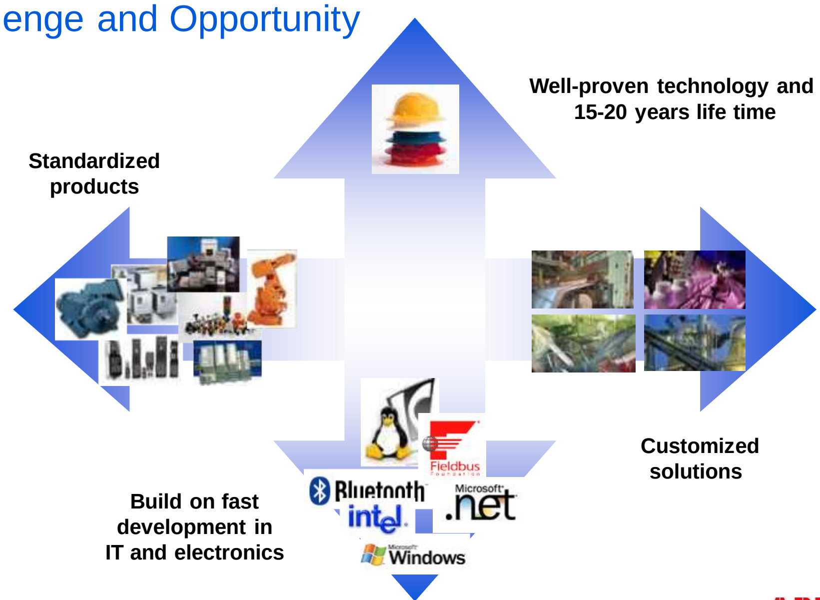
Software at ABB
Integration Platform Architecture



Software at ABB
Solution Approach Using Enterprise Processes



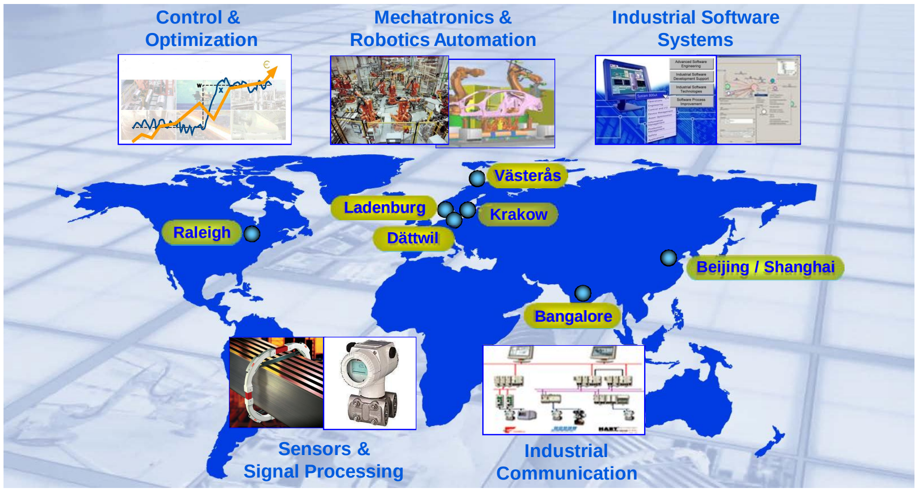
Software at ABB
Challenge and Opportunity



Design Challenges in Automation & Power Domains (for Hardware and Software)

- **Safety and security**
  - E.g. Security guidelines from North American Electric Reliability Corporation's (NERC)
  - E.g. Stuxnet threat (for entire industry)
- **Remote locations**
  - Unmanned plants
  - Extreme environmental conditions
- **Diversity and lifetime of installed base**
  - 1000s of products
  - Some of them 40+ years old; news ones to last long
- **Technology evolution (and debt)**
  - Operating systems
  - WWW, TCP/IP, Ethernet

ABB Corporate Research Centers
Automation Research Programs



ABB

Industrial Software Systems (ISS) Program


Sitemap Anmeldung

ABB Power and productivity for a better world™

ABB in der Schweiz

Home Über uns Produkte und Dienstleistungen Medien Job und Karriere ABB Group

Technology

- Technology
- + Global lab power
- − Global lab automation
  - Control and optimization
  - Industrial communication
  - **Industrial Software Systems**
  - Mechatronics
  - Sensors and signal processing
- + Corporate research centers
- + Technology Ventures
- + Publications
- + Select technologies
- Cyber security
- DC power

Industrial Software Systems Program

The ABB Corporate Research program Industrial Software Systems (ISS) supports ABB product development units with world-class expertise in information/software technology, software engineering, and computer science in making effective use of novel information technologies and in applying state-of-the-art software engineering methods in order to build products that are attractive to ABB customers.

The Industrial Software Systems program fulfills its mission of ensuring ABB competitiveness and technology leadership through projects in four research areas:

- **Software engineering** creates methods and tools to increase quality and reduce cost of ABB internal software development.
- **Software architecture** researches new ways of designing, building, and maintaining complex software systems over multi-decade lifecycles.
- **IT systems architecture** explores the usage of secure and highly scalable HW/SW systems composed of ABB as well as COTS/FOSS subsystems for information processing and analysis.
- **Human-computer interaction** investigates user experience and situational awareness in the context of industrial systems.

The ISS program team in five locations worldwide (Västeras/SE, Raleigh/US, Bangalore/IN, Ladenburg/DE, Baden/CH), works with software development units across all ABB divisions and geographic areas. Interdisciplinary projects are executed in collaboration with other ABB Corporate Research programs as well as partners in universities.

Last edited 2012-01-10

Suche →OK

- + Seite bewerten
- + Diese Seite empfehlen

Kontakt

→ Software research contact

Links

- → ABB Corporate Research Center locations
- → ABB Software Research Grants
- → Software engineering research
- → Security research
- → ISS publications
- → Open positions

http://www.abb.com/softwareresearch

ABB

Agenda

- Context
  - Software and software research at ABB
  - Distributed process control systems
  - Domain-specific design challenges
- **Sample projects and initiatives**
  - Q-ImPrESS performance modeling
  - Software development improvement initiatives
- Architectural Knowledge Management (AKM) practices
  - Decision rationale – past and present
  - Capturing advice – relevant issues, good justifications
  - Towards an sustainability guide for AKM
- Summary and roadmap

Motivation
Problems of Software Evolution at ABB

- Continuous *evolution* of ABB software systems
  - New requirements, technologies, failure reports

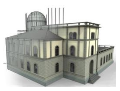
➔ Software maintenance and evolution
are a large cost factor for ABB software development

- Current practice
  - *Experience* to rationalize design decisions
  - *Prototyping* for new technologies, performance impacts
  - *Unknown* change impacts on performance/reliability

➔ Apply model-based prediction methods for systematic decision support to save costs and achieve higher quality?

Quality Impact Predictions for Evolving Service-Oriented Systems (Q-ImPreSS)

Manual Model Creation



Modelling static structure

- Analyzed architectural documenation
- Identified four key use cases
- Abstraction level: process = component

Modelling dynamic structure

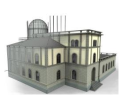
- Created testbed, installed system
- Recorded component transitions
- Derived transition probabilities

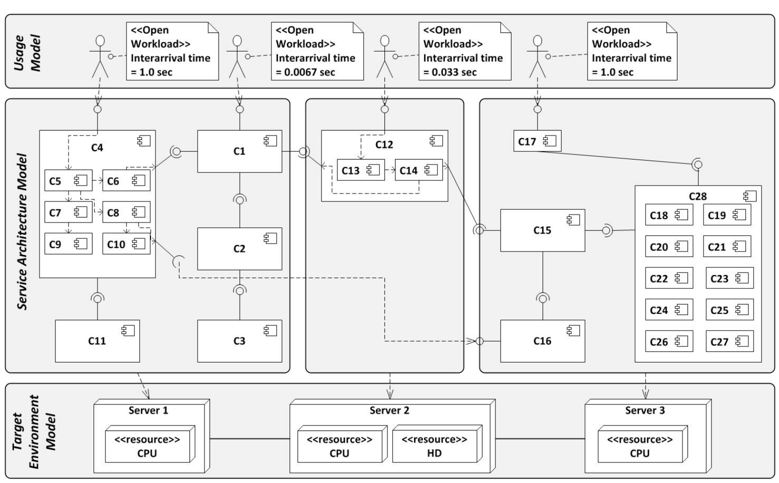
Validating the model

- Created Q-ImPrESS model in workbench
- Applied Q-ImPrESS consistency checker
- Discussed the model with architects

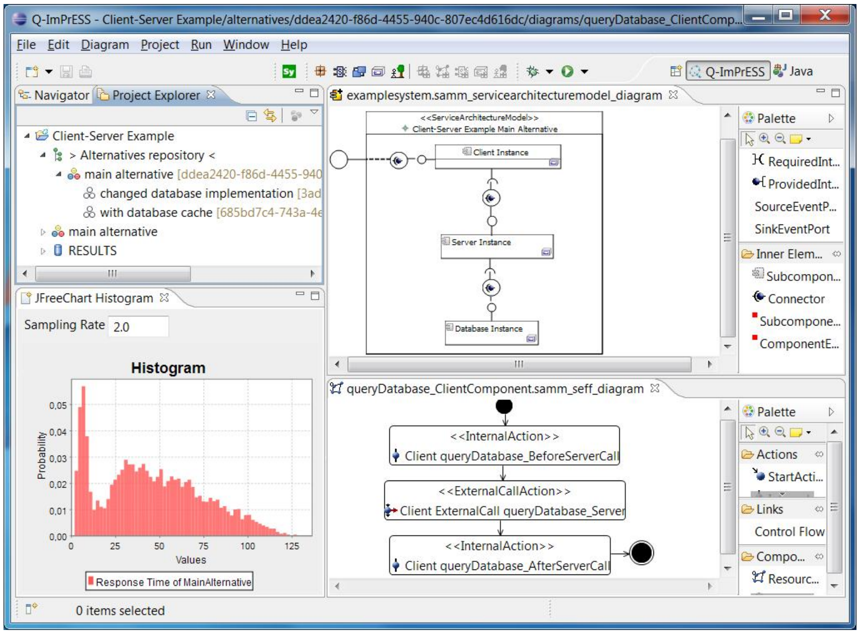
Q-ImPrESS Model of an ABB Process Control System

Manual Model Creation



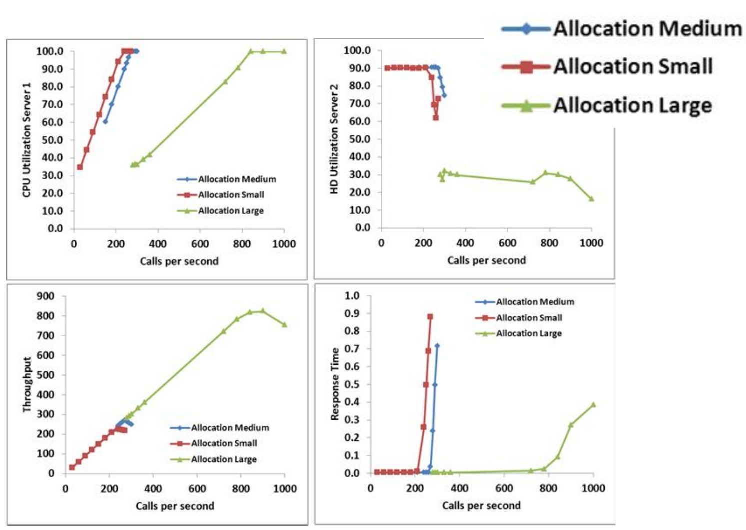


Q-ImPrESS Workbench



Performance Prediction

Sample Predictions for Different Design Alternatives



Performance Prediction
Results: Measurements vs. Simulation Results

Workload	PerfMon Measured	SimuCom Prediction	Error (%)	LQNS Prediction	Error (%)
30	17.146	12.467	27.288	12.464	27.305
60	26.681	22.366	16.174	22.343	16.260
90	31.902	32.347	1.395	32.322	1.317
120	39.016	42.432	8.754	42.329	8.490
150	51.929	51.943	0.027	51.760	0.326

Pro

- Achieved prediction error below 30 percent
- Easy to analyze different evolution scenarios

Con

- Data collection consumed more time than expected
- Many bottlenecks below the architectural level

Q-Impress – Many Lessons Learned
Results Presented at ICSE 2011 (URL)

- Q-ImPrESS:
  - Provides a structured method and useful tool support
  - Is best used for evolutionary changes, not full redesigns
  - Still needs to demonstrate costs/benefits
- Future work:
  - More robust reverse engineering tools
  - Model transformations from UML to Q-ImPrESS
  - Tools and best practices for data collection

➔ **More research and tool development needed**

http://www.infoq.com/news/2011/04/palladio_tool

Agenda

- Context
  - Software and software research at ABB
  - Distributed process control systems
  - Domain-specific design challenges
- Sample projects and initiatives
  - Q-ImPrESS performance modeling
  - Software development improvement initiatives
- **Architectural Knowledge Management (AKM) practices**
  - Decision rationale – past and present
  - Capturing advice – relevant issues, good justifications
  - Towards an sustainability guide for AKM
- Summary and roadmap

Architectural Knowledge Management (AKM) (1/2)

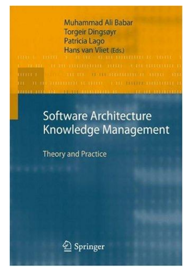
- D. Perry/A. Wolf (1992): *Software Architecture = {Elements, Form, Rationale}*
- P. Kruchten (2004): presentation and workshop paper (QoSA 2006 update)

> **Ontology of Architectural Decisions** [Endnotes][Bibtex]
>
> More than just lots of UML diagrams, architectural knowledge is embodied in the dozen of architectural design decisions, and their numerous relationships. This paper presents a taxonomy of such decisions.
>
> Philippe Kruchten, "An Ontology of Architectural Design Decisions," in: Jan Bosch (ed.), *Proc. of the 2nd Workshop on Software Variability Management*, Groningen, NL, Dec. 3-4, 2004
>
> Read the Article »

- A. Jansen, J. Bosch: Software Architecture as a Set of Architectural Design Decisions (2005)
  - An architectural (design) decision is *“a description of the set of architectural additions, subtractions and modifications to the software architecture, the rationale, and the design rules, design constraints and additional requirements that (partially) realize one or more requirements on a given architecture.“*
  - Rationale defined as: *“The reasons behind an architectural design decision are the rationale of an architectural design decision. It describes why a change is made to the software architecture.“*

Architectural Knowledge Management (AKM) (2/2)

- SHARK workshops and WICSA/QoSA conference tracks since 2006
  - *Architectural Knowledge = Architecture Design + Architectural Decisions* (P. Kruchten, P. Lago, H. van Vliet, QoSA 2006)
- Special Issue: Architectural Decisions and Rationale, Journal of Systems and Software 82(8), 2009 (editorial and four papers, e.g., one from presenter)
- Book: Software Architecture Knowledge Management – Theory and Practice, Springer (2009)
  - Management strategies – *explicit* vs. *implicit*
  - Use cases, ontologies (e.g. Griffin core model), links to other design artifacts
  - Tool survey (research prototypes)
  - Case studies (e.g. SOA reference architecture with recurring architectural decisions from IBM)
- IEEE 42010 now makes decision capturing mandatory (2011)



ABB

An Example of an Architectural Decision (Modeled in Sparx Systems Enterprise Architect)

Unique identifier: AD-01, **Status:** approved, **Owner:** cholzim

Link to system concerns
See architectural principles stated on the enterprise level: IT security requirements

Rationale
In the context of the historian component, facing the data privacy requirements specified in the corporate security guidelines, we decided to encrypt the persistent storage to achieve confidentiality.

Constraints and assumptions:
Performance is good enough, certificate management can be handled.

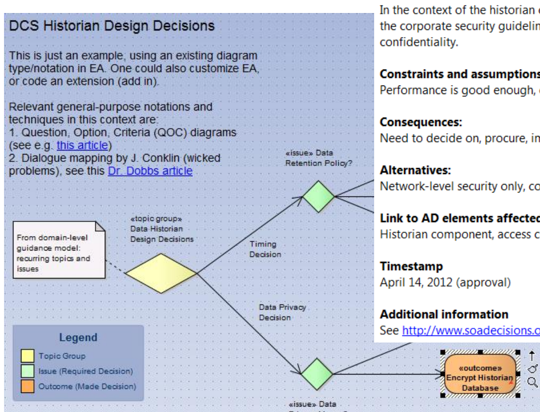
Consequences:
Need to decide on, procure, implement some crypto library and/or hardware (TBD)

Alternatives:
Network-level security only, combined with role-based access control

Link to AD elements affected by the decision:
Historian component, access channel, security subsystem

Timestamp
April 14, 2012 (approval)

Additional information
See http://www.soadecisions.org



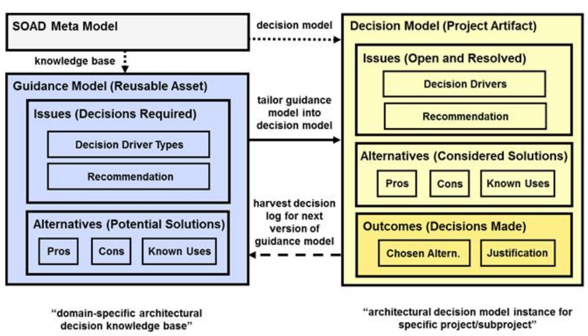
ABB

Many Metamodels – Few Models

- Many *metamodels and templates* have been published
  - IBM Unified Method Framework (since 1998), see SATURN 2010 session
  - Key decision template suggested by Bredemeyer Consulting
  - Table by J. Tyree/A. Akerman in IEEE Software 22(2), 2005
  - arc42 also suggests a template, e.g. wiki-style (Germany/Austria)
  - TOGAF 9 has the notion of an architectural contract (with rationale)
- But only very few *models* are publicly available (confidentiality/maintenance?)
  - The seminal book by M. Shaw and D. Garlan (1996) features a partial design space for user interface architectures (Chapter 5, page 97)
  - Perspectives on Web Services (2003) has 26 recurring decisions
  - Informal coverage in integration patterns book and SWEBOK (2004)
  - Guidance model for SOA partially published in PhD thesis and tutorials

ABB

SOAD (2006-2011): Generic Metamodel



Source: O. Zimmermann, Architectural Decisions as Reusable Design Assets.
IEEE Software, vol. 28, no. 1, pp. 64-69, Jan./Feb. 2011.

- Existing metamodels and templates refatcored and extended for reuse
  - Before: documentation – after the fact
  - Now: design guidance – forward looking

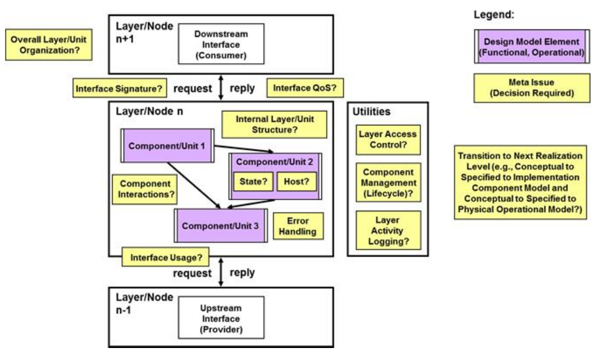
http://soadecisions.org/soad.htm

ABB

Sample Model Content for SCADA/DCS Historian Sources: Domain Patterns and Recurring Issues

- Conceptual issues:
  - Data point selection (granularity, sampling rate)?
  - Data retention policy (duration, protection)?
  - Database style (flat file/relational/document-oriented)?
- Technology issues:
  - Query language?
  - Remoting protocol?
  - Encryption protocol?
- Vendor asset issues:
  - OS, MW, HW choices (make or buy)?
  - Implementation providers for selected technologies?
  - Backup and restore system?

SOAD (2006-2011): Recurring Issues



Source: O. Zimmermann, Architectural Decisions as Reusable Design Assets. IEEE Software, vol. 28, no. 1, pp. 64-69, Jan./Feb. 2011.

- Patterns + meta issues yield *guidance models* for a domain
  - Successfully applied to Service-Oriented Architecture (SOA) Design, cloud computing, strategic outsourcing
- Meta issue catalog organized by layer/node type, by component/connector

Recurring Issues (1/2)

Artifact	Decision Topic	Recurring Issues (Decisions Required)
Enterprise architecture documentation [SZ92, ZTP03]	IT strategy	Buy vs. build strategy, open source policy
	Governance	Methods (processes, notations), tools, reference architectures, coding guidelines, naming standards, asset ownership
System context [CCS07]	Project scope	External interfaces, incoming and outgoing calls (protocols, formats, identifiers), service level agreements, billing
Other viewpoints [Kru95]	Development process	Configuration management, test cases, build/test/production environment staging
	Physical tiers	Locations, security zones, nodes, load balancing, failover, storage placement
	Data management	Data model reach (enterprise-wide?), synchronization/replication, backup strategy
Architecture overview diagram [Fow03, CCS07]	Logical layers	Coupling and cohesion principles, functional decomposition (partitioning)
	Physical tiers	Locations, security zones, nodes, load balancing, failover, storage placement
	Data management	Data model reach (enterprise-wide?), synchronization/replication, backup strategy
Architecture overview diagram [Eva03, Fow03]	Presentation layer	Rich vs. thin client, multi-channel design, client conversations, session management
	Domain layer (process control flow)	How to ensure process and resource integrity, business and system transactionality
	Domain layer (remote interfaces)	Remote contract design (interfaces, protocols, formats, timeout management)
	Domain layer (component-based development)	Interface contract language, parameter validation, Application Programming Interface (API) design, domain model
	Resource (data) access layer	Connection pooling, concurrency (auto commit?), information integration, caching
	Integration	Hub-and-spoke vs. direct, synchrony, message queuing, data formats, registration

Source: O. Zimmermann, An architectural decision modeling framework for service oriented architecture design. PhD thesis, Stuttgart University, 2009.

ABB

Recurring Issues (2/2)

Artifact	Decision Topic	Recurring Issues (Decisions Required)
Logical component [ZTP03]	Security	Authentication, authorization, confidentiality, integrity, non-repudiation, tenancy
	Systems management	Fault, configuration, accounting, performance, and security management
Logical component [ZZG+08]	Lifecycle management	Lookup, creation, static vs. dynamic activation, instance pooling, housekeeping
	Logging	Log source and sink, protocol, format, level of detail (verbosity levels)
	Error handling	Error logging, reporting, propagation, display, analysis, recovery
Components and connectors [ZTP03, CCS07]	Implementation technology	Technology standard version and profile to use, deployment descriptor settings (QoS)
	Deployment	Collocation, standalone vs. clustered
Physical node [YRS+99]	Capacity planning	Hardware and software sizing, topologies
	Systems management	Monitoring concept, backup procedures, update management, disaster recovery

Source: O. Zimmermann, An architectural decision modeling framework for service oriented architecture design. PhD thesis, Stuttgart University, 2009.

ABB

How Much Design Rationale is Enough?

- Little information what/how much to capture:
  - Most metamodels and templates ask for a lot of detail (cost/benefit?)
  - G. Fairbanks suggests a lean/minimalistic approach to rationale capturing in his architectural haikus (presented at WICSA 2011):
    - *Requirement <driver-x> is a priority, so we chose design <alt-y>, accepting downside <consequence-z>*

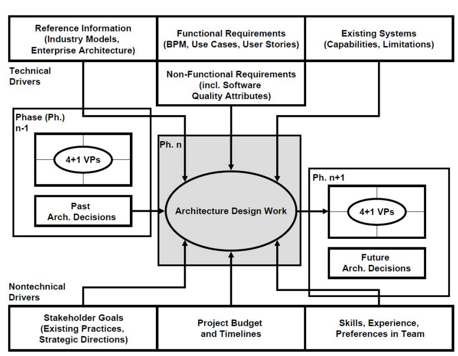
(WH)Y?

  - My version (the Y-approach):
    - *In the context of <use case/user story u>, facing <concern c>, we decided for <option o> to achieve <quality q>*
    - These Y statements yield a bullet list of open/closed (design) issues (link to project management!)
    - Can go to appendix of software architecture document, Wiki, spreadsheet, notes attached to UML model elements

ABB

Valid and Invalid Justifications
Food for Architectural Evaluations/Reviews!

- **Convincing rationale:**
  - Direct link to requirements (the "Y")
    - Quality attributes in particular, but also functional requirements and constraints
  - Positive experience on previous project
    - Or prototype, experiment, simulation
  - Existing skills, license agreements
    - Other project management concerns
- **Poor justifications:**
  - Market momentum
    - Technology or vendor push
  - Only one alternative known/considered
    - Other killer phrases
  - Keep CVs of team members current



Source: O. Zimmermann, An architectural decision modeling framework for service oriented architecture design. PhD thesis, Stuttgart University, 2009.

ABB

Good and Bad Justifications, Part 1

Decision driver type	Valid justification	Counter example
Wants and needs of external stakeholders	Alternative A best meets user expectations and functional requirements as documented in user stories, use cases, and business process model.	End users want it, but no evidence for a pressing business need. Technical project team never challenged the need for this feature. Technical design is prescribed in the requirements documents.
Architecturally significant requirements	Nonfunctional requirement XYZ has higher weight than any other requirement and must be addressed; only alternative A meets it.	Do not have any strong requirements that would favor one of the design options, but alternative B is the market trend. Using it will reflect well on the team.
Conflicting decision drivers and alternatives	Performed a trade-off analysis, and alternative A scored best. Prototype showed that it's good enough to solve the given design problem and has acceptable negative consequences.	Only had time to review two design options and did not conduct any hands-on experiments. Alternative B does not seem to perform well, according to information online. Let's try alternative A.

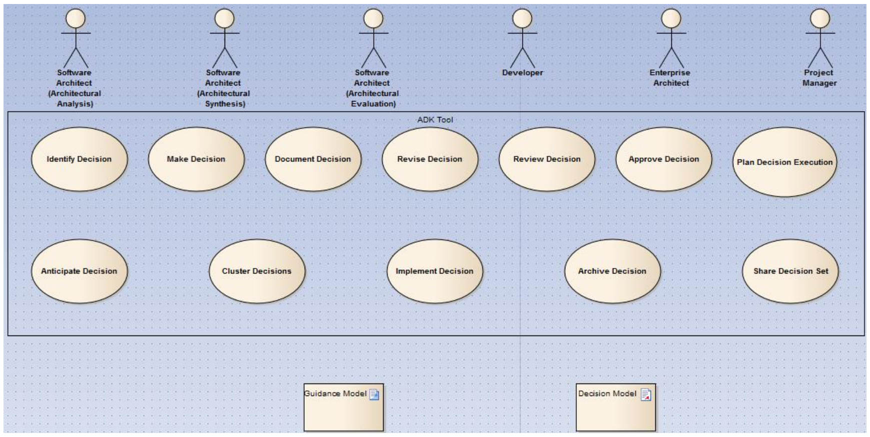
Source: Zimmermann O., Schuster N., Eeles P., Modeling and Sharing Architectural Decisions, Part 1: Concepts. IBM developerWorks, 2008

Good and Bad Justifications, Part 2

Decision driver type	Valid justification	Counter example
Reuse of an earlier design	Facing the same or very similar NFRs as successfully completed project XYZ. Alternative A worked well there. A reusable asset of high quality is available to the team.	We've always done it like that. Everybody seems to go this way these days; there's a lot of momentum for this technology.
Prefer do-it-yourself over commercial off-the-shelf (build over buy)	Two cornerstones of our IT strategy are to differentiate ourselves in selected application areas, and remain master of our destiny by avoiding vendor lock-in. None of the evaluated software both meets our functional requirements and fits into our application landscape. We analyzed customization and maintenance efforts and concluded that related cost will be in the same range as custom development.	Price of software package seems high, though we did not investigate total cost of ownership (TCO) in detail. Prefer to build our own middleware so we can use our existing application development resources.
Anticipation of future needs	Change case XYZ describes a feature we don't need in the first release but is in plan for next release. Predict that concurrent requests will be x per second shortly after global rollout of the solution, planned for Q1/2009.	Have to be ready for any future change in technology standards and in data models. All quality attributes matter, and quality attribute XYZ is always the most important for any software-intensive system.

Source: Zimmermann O., Schuster N., Eeles P., Modeling and Sharing Architectural Decisions, Part 1: Concepts. IBM developerWorks, 2008

Sustainable Architectural Decision Knowledge
Wanted: Integrated Decision/Design Tool Chain



- Tool builders should justify capture their design decisions (like any architect)… and share them with their collaborators!

Agenda

- Context
  - Software and software research at ABB
  - Distributed process control systems
  - Domain-specific design challenges
- Sample projects and initiatives
  - Q-ImPrESS performance modeling
  - Software development improvement initiatives
- Architectural Knowledge Management (AKM) practices
  - Decision rationale – past and present
  - Capturing advice – relevant issues, good justifications
  - Towards an sustainability guide for AKM
- **Summary and roadmap**

Summary and Conclusions

- Software and software architecture play a key role at ABB
  - Projects, programs, initiatives in place
  - Some key themes: modeling, reuse, rationale, sustainability
- Architecture design is driven both by functional and by non-fucntional requirements – and constraints of both kinds
  - Design techniques and modeling tools should combine these
  - Hard to see the forest for the trees – guidance required
- Answers to why questions matter – and tend to be more sustainable than most component-and-connector diagrams (reuse of know how!)
  - Explicit knowledge management does not imply big design upfront, evolutionary architectures and/designs should be justified
    - See recently released IEEE 42010 standard
  - *Try a rationale haiku and/or the “Y“ (FR+NFR->Justified Outcome)* to document the essence of a decision

ABB

Recommended Reading

- Jansen/Bosch, *Software Architecture as a Set of Architectural Design Decisions*, WICSA 2005. 5th Working IEEE/IFIP Conference on Software Architecture
  - Motivation for decision capturing, basic definitions
- ISO/IEC/IEEE 42010:2011, *Systems and software engineering—Architecture description*.
  - Rationale – what to capture, how to capture
- Zimmermann O., *Architectural Decisions as Reusable Design Assets*. IEEE Software, vol. 28, no. 1, pp. 64-69, Jan./Feb. 2011
  - From documentation to design guidance, SOA examples
- Hollender, *Collaborative Process Automation Systems*, ISA 2010
  - Domain-specific quality attributes and decisions

ABB

Power and productivity
for a better world™