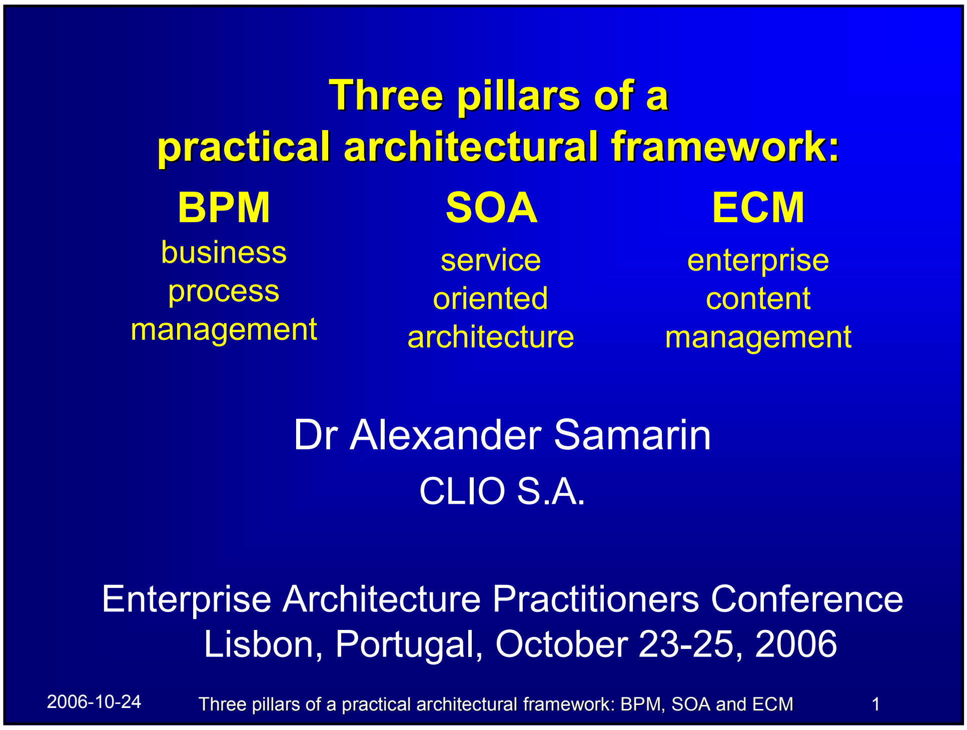


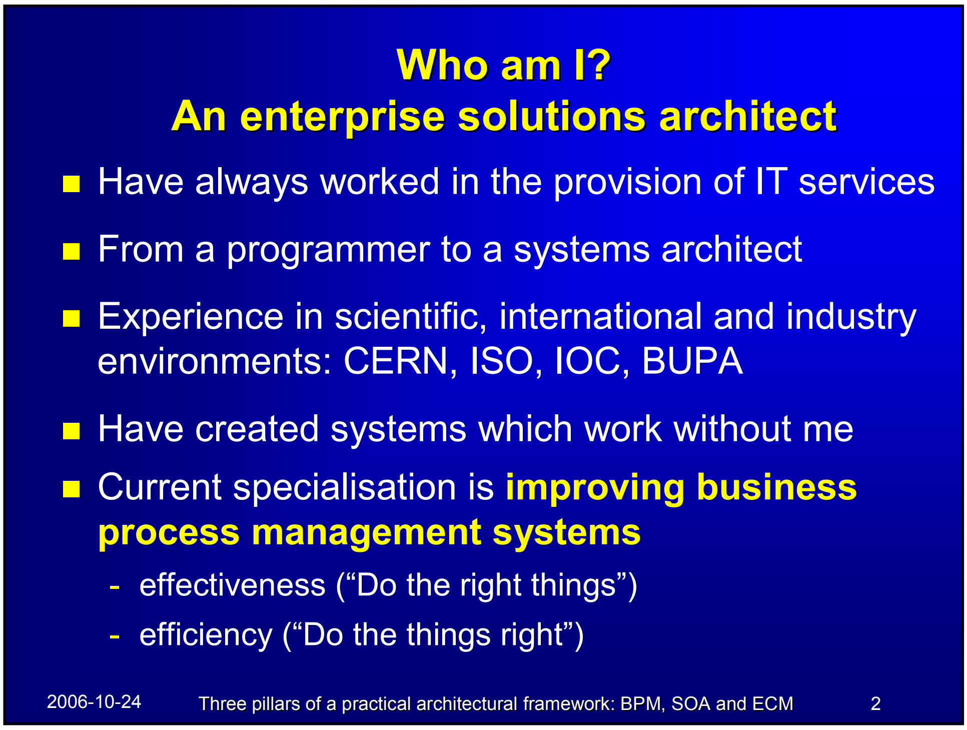
Three pillars of a practical architectural framework:

BPM	**SOA**	**ECM**
business process management	service oriented architecture	enterprise content management

Dr Alexander Samarin

CLIO S.A.

Enterprise Architecture Practitioners Conference
Lisbon, Portugal, October 23-25, 2006



Who am I?
An enterprise solutions architect

- Have always worked in the provision of IT services
- From a programmer to a systems architect
- Experience in scientific, international and industry environments: CERN, ISO, IOC, BUPA
- Have created systems which work without me
- Current specialisation is **improving business process management systems**
  - effectiveness (“Do the right things”)
  - efficiency (“Do the things right”)

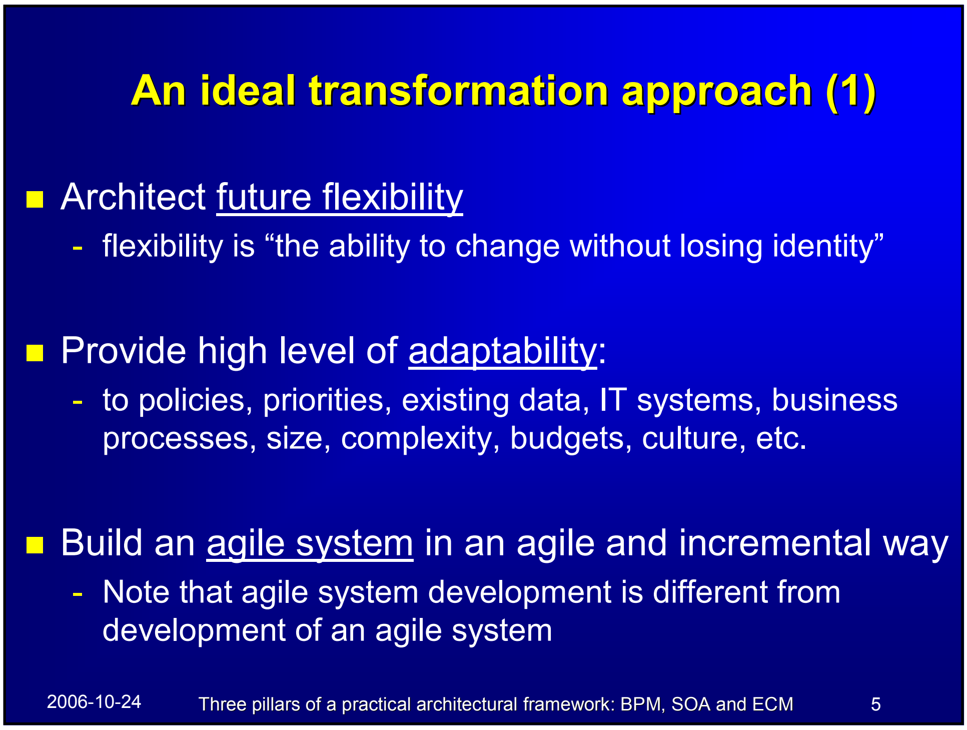
Today's business opportunity: better architecture of enterprise systems

- No architecture blueprint
- 38 years of construction
- 160 rooms, 497 doorways, 950 doors
- Over 20 tonnes of paint required

- No disrupting of river traffic activities
- The committee evaluated 50 projects
- Three architectural techniques
- 8 years of construction
- Modernised for new technology

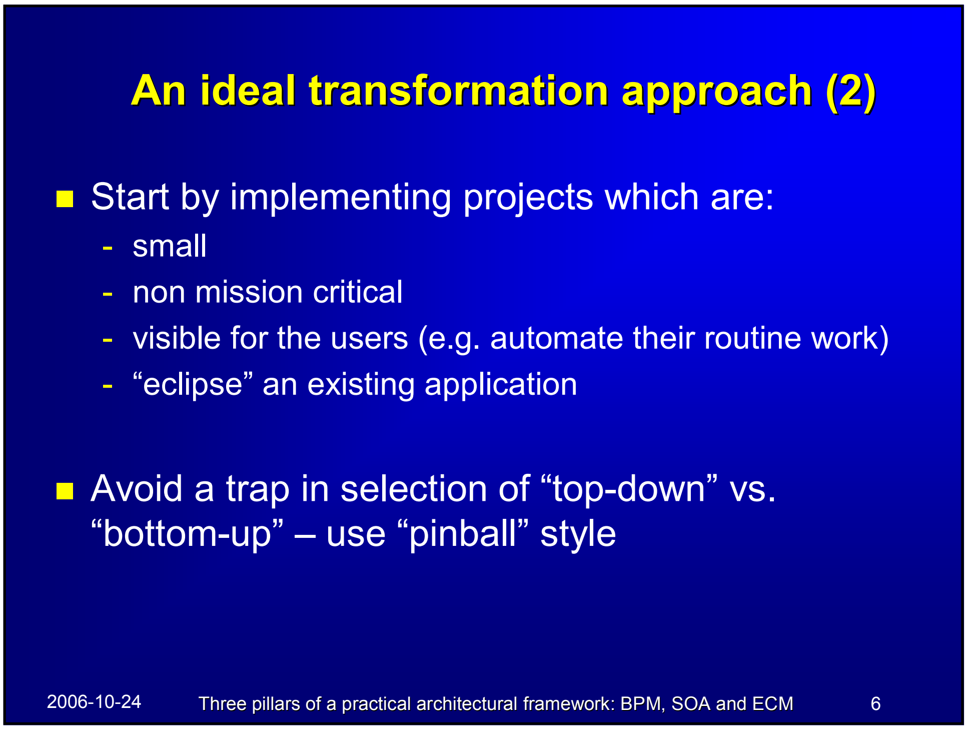
A daunting optimisation task

- From a typical enterprise environment:
  - a complex system of systems that has grown over years
  - a very hostile environment for new things
- To a coherent, smoothly evolving, business system
  - easy to maintain and develop further
- Subject to socio-technical aspects:
  - **how** you do something is sometimes more important than **what** you do



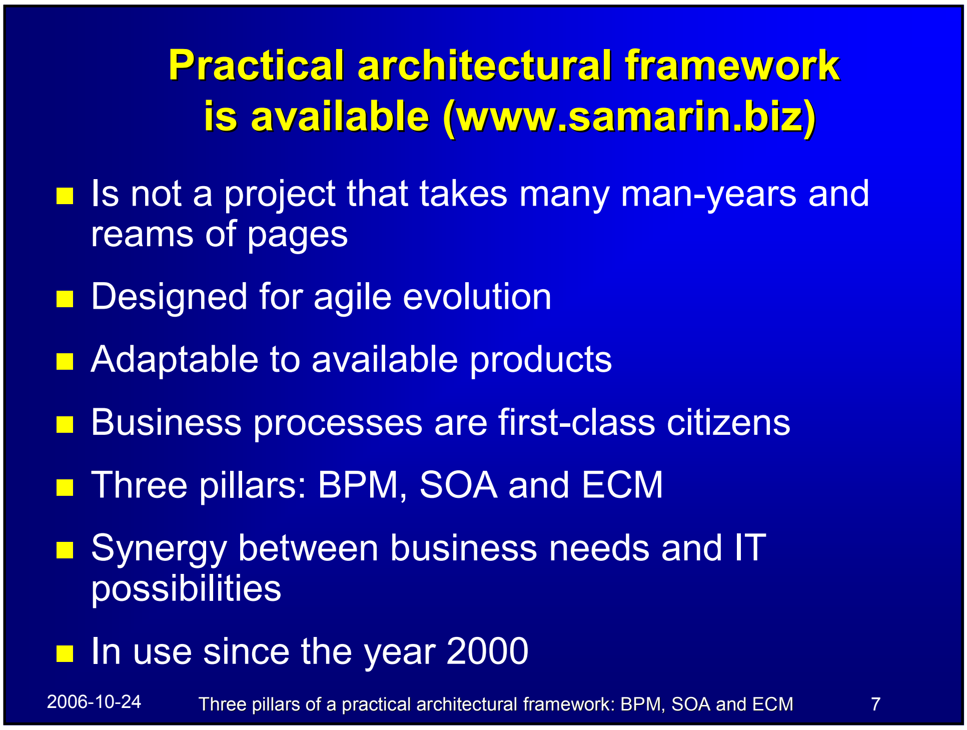
An ideal transformation approach (1)

- Architect future flexibility
  - flexibility is “the ability to change without losing identity”
- Provide high level of adaptability:
  - to policies, priorities, existing data, IT systems, business processes, size, complexity, budgets, culture, etc.
- Build an agile system in an agile and incremental way
  - Note that agile system development is different from development of an agile system



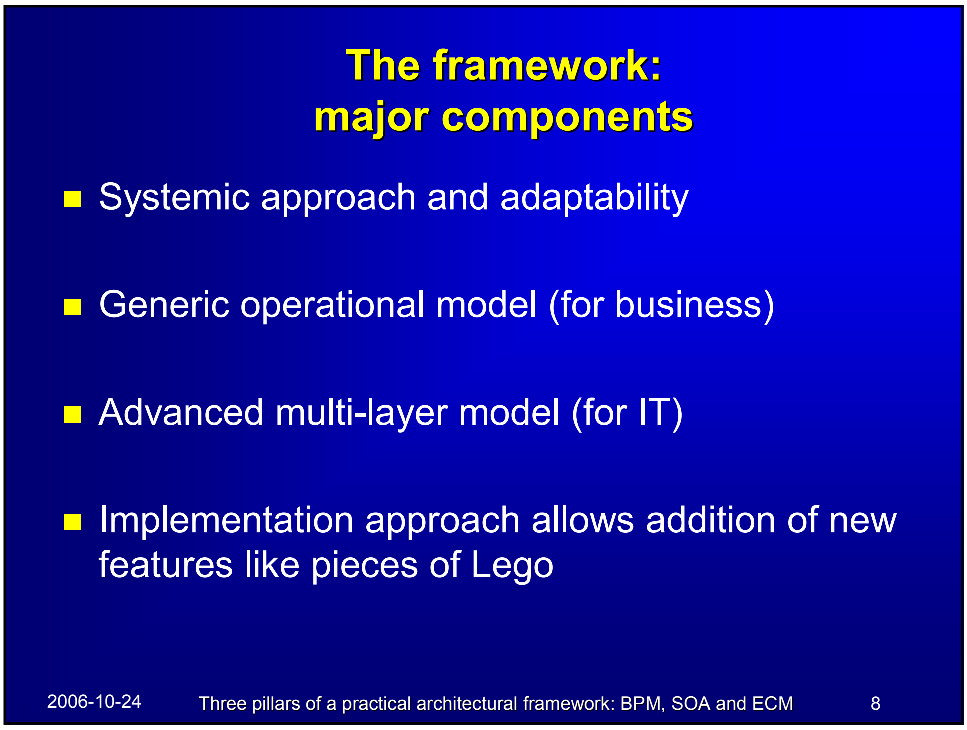
An ideal transformation approach (2)

- Start by implementing projects which are:
  - small
  - non mission critical
  - visible for the users (e.g. automate their routine work)
  - “eclipse” an existing application
- Avoid a trap in selection of “top-down” vs. “bottom-up” – use “pinball” style



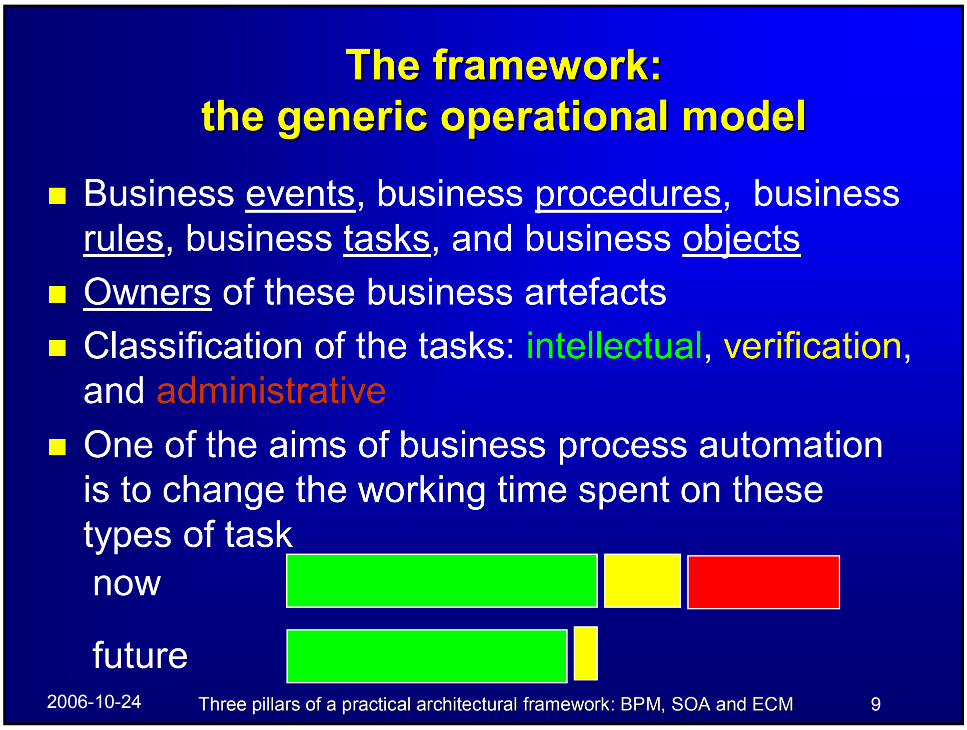
Practical architectural framework is available (www.samarin.biz)

- Is not a project that takes many man-years and reams of pages
- Designed for agile evolution
- Adaptable to available products
- Business processes are first-class citizens
- Three pillars: BPM, SOA and ECM
- Synergy between business needs and IT possibilities
- In use since the year 2000



The framework: major components

- Systemic approach and adaptability
- Generic operational model (for business)
- Advanced multi-layer model (for IT)
- Implementation approach allows addition of new features like pieces of Lego

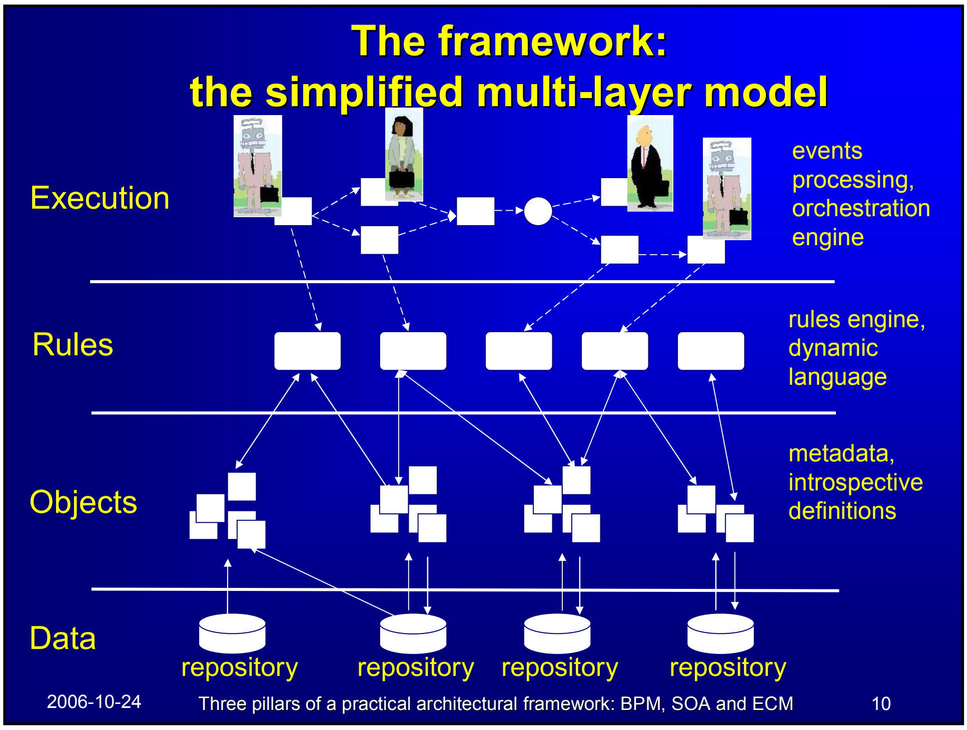


The framework: the generic operational model

- Business events, business procedures, business rules, business tasks, and business objects
- Owners of these business artefacts
- Classification of the tasks: intellectual, verification, and administrative
- One of the aims of business process automation is to change the working time spent on these types of task

now

future



The framework: the simplified multi-layer model

Execution

events processing, orchestration engine

Rules

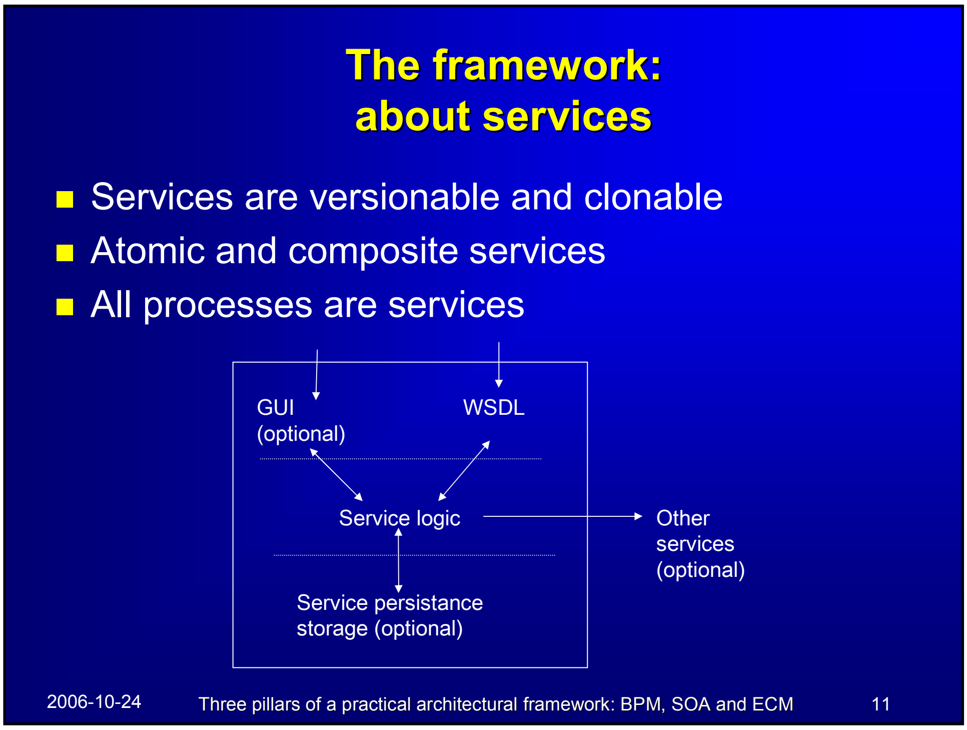
rules engine, dynamic language

Objects

metadata, introspective definitions

Data

repository repository repository repository



The framework: about services

- Services are versionable and clonable
- Atomic and composite services
- All processes are services

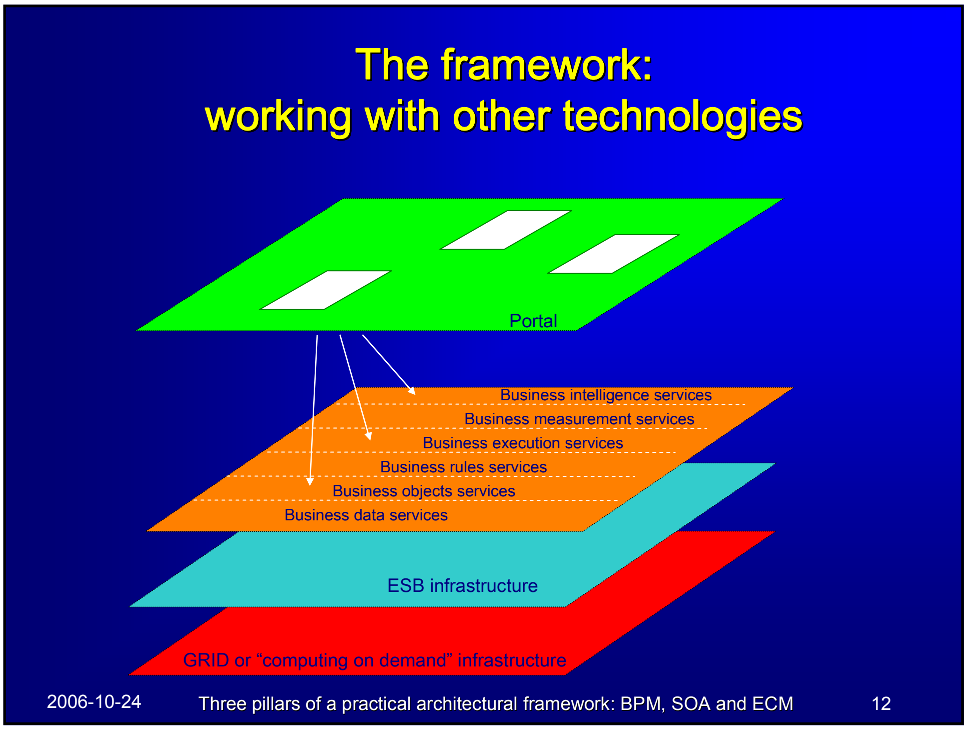
GUI (optional)

WSDL

Service logic

Other services (optional)

Service persistance storage (optional)



The framework: working with other technologies

Portal

Business intelligence services

Business measurement services

Business execution services

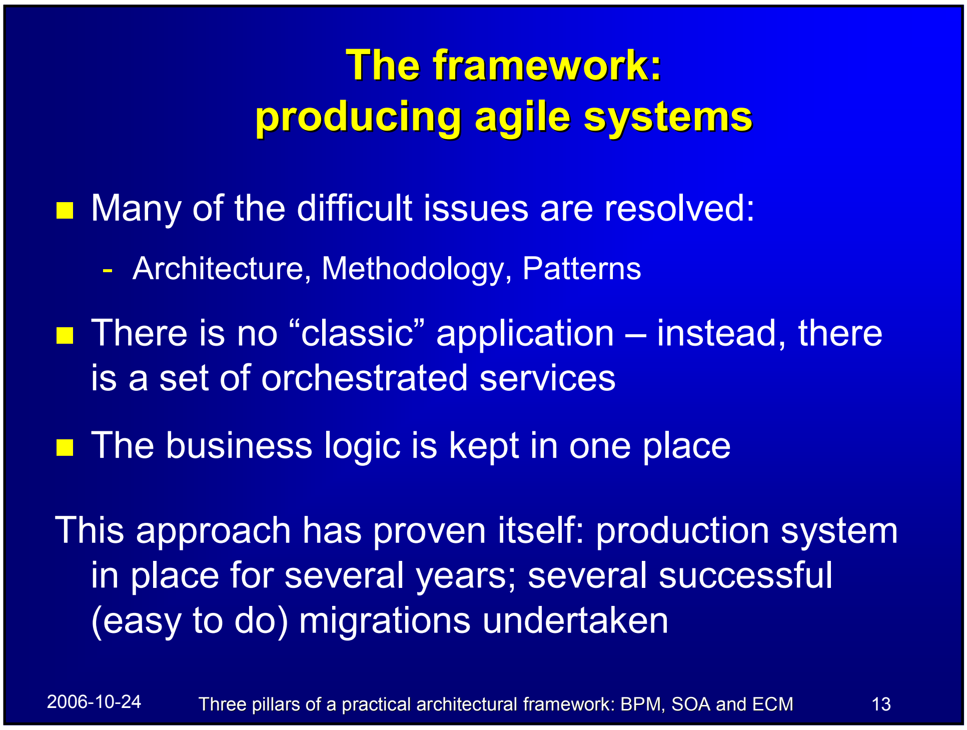
Business rules services

Business objects services

Business data services

ESB infrastructure

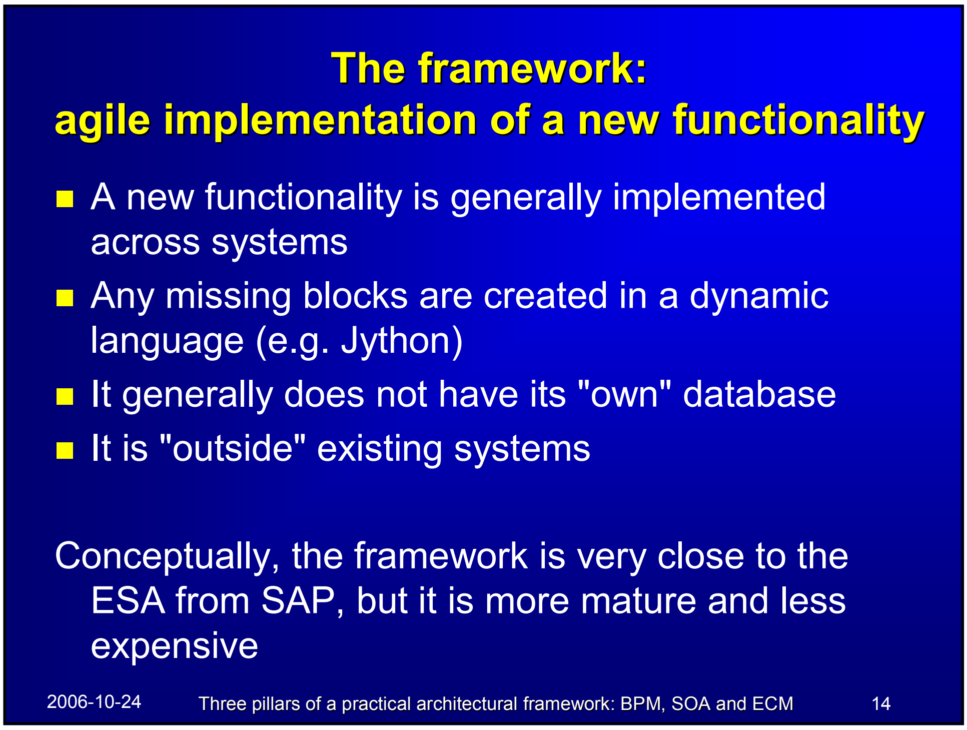
GRID or “computing on demand” infrastructure



The framework: producing agile systems

- Many of the difficult issues are resolved:
  - Architecture, Methodology, Patterns
- There is no “classic” application – instead, there is a set of orchestrated services
- The business logic is kept in one place

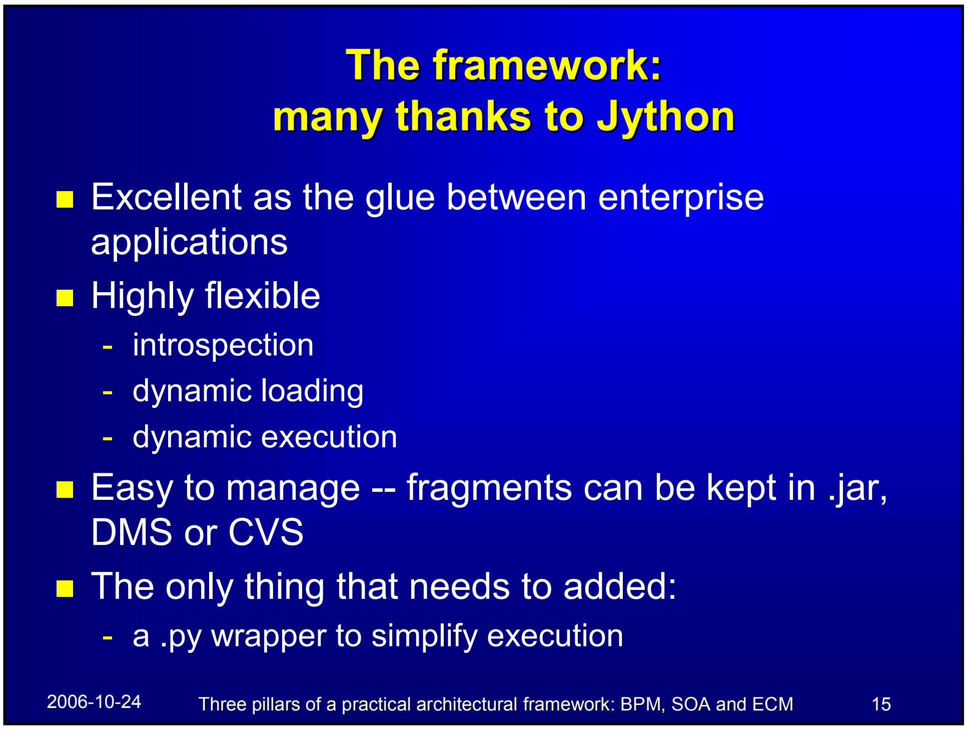
This approach has proven itself: production system in place for several years; several successful (easy to do) migrations undertaken



The framework: agile implementation of a new functionality

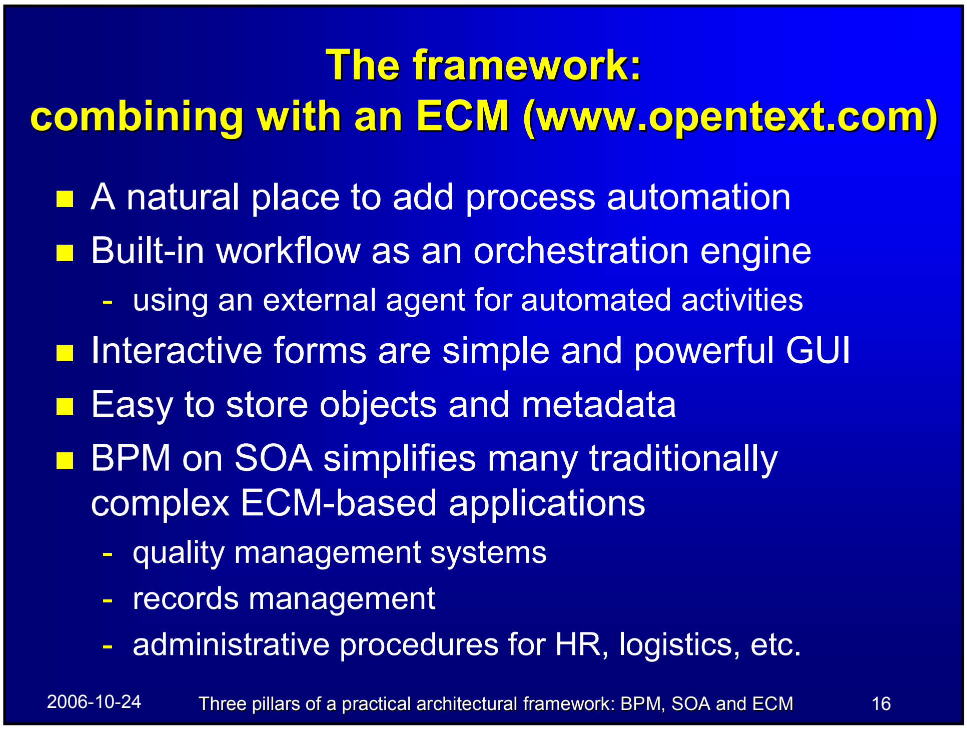
- A new functionality is generally implemented across systems
- Any missing blocks are created in a dynamic language (e.g. Jython)
- It generally does not have its "own" database
- It is "outside" existing systems

Conceptually, the framework is very close to the ESA from SAP, but it is more mature and less expensive



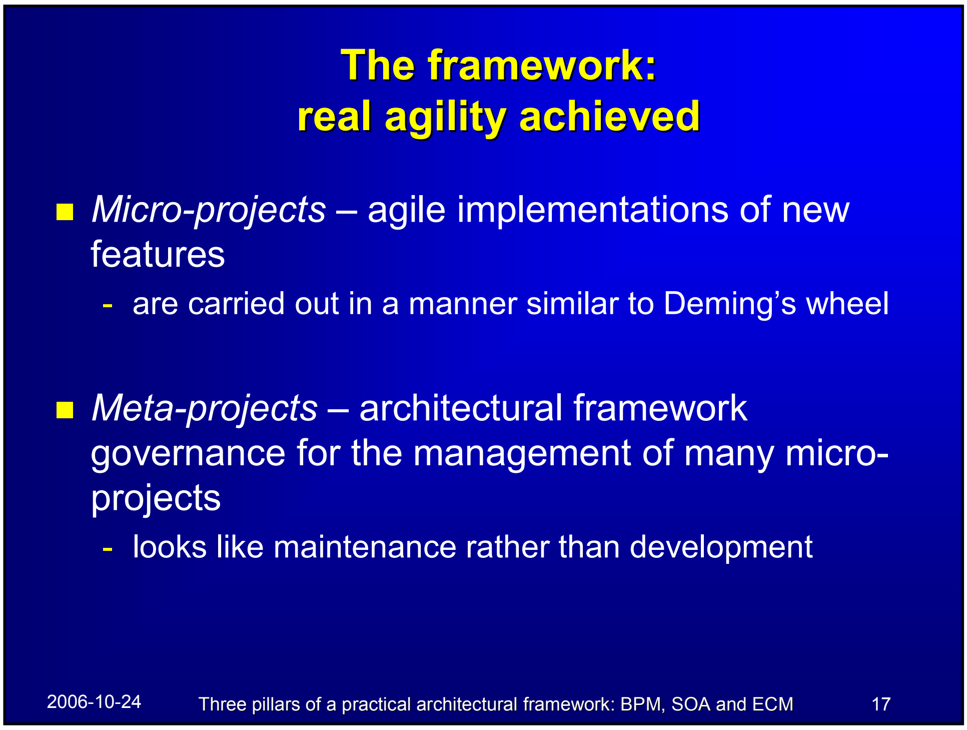
The framework: many thanks to Jython

- Excellent as the glue between enterprise applications
- Highly flexible
  - introspection
  - dynamic loading
  - dynamic execution
- Easy to manage -- fragments can be kept in .jar, DMS or CVS
- The only thing that needs to added:
  - a .py wrapper to simplify execution



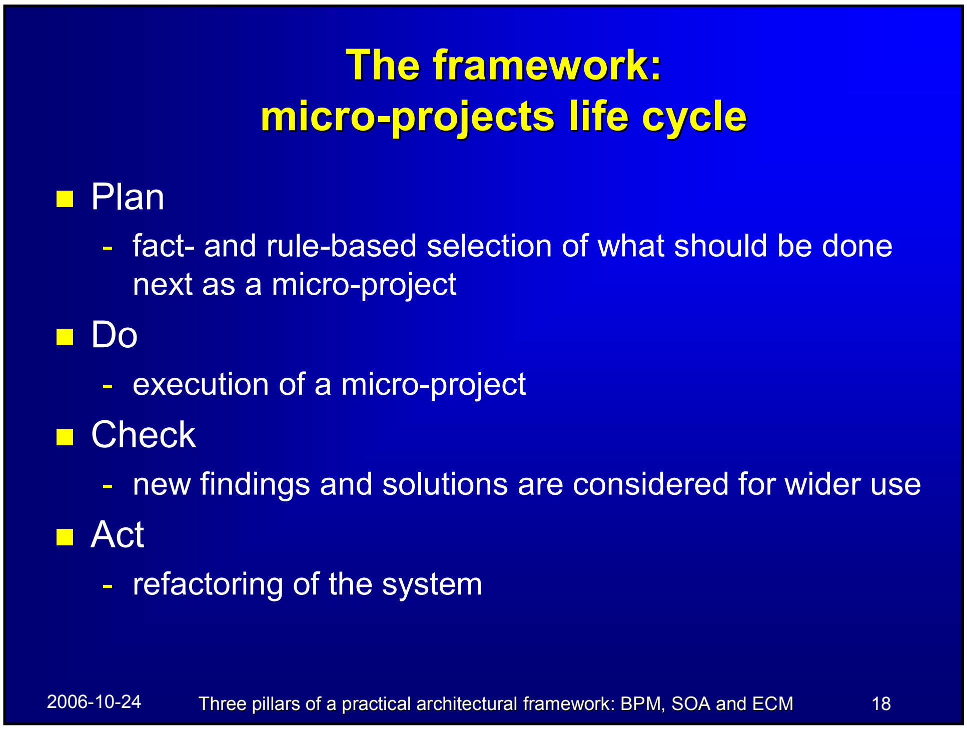
The framework: combining with an ECM (www.opentext.com)

- A natural place to add process automation
- Built-in workflow as an orchestration engine
  - using an external agent for automated activities
- Interactive forms are simple and powerful GUI
- Easy to store objects and metadata
- BPM on SOA simplifies many traditionally complex ECM-based applications
  - quality management systems
  - records management
  - administrative procedures for HR, logistics, etc.



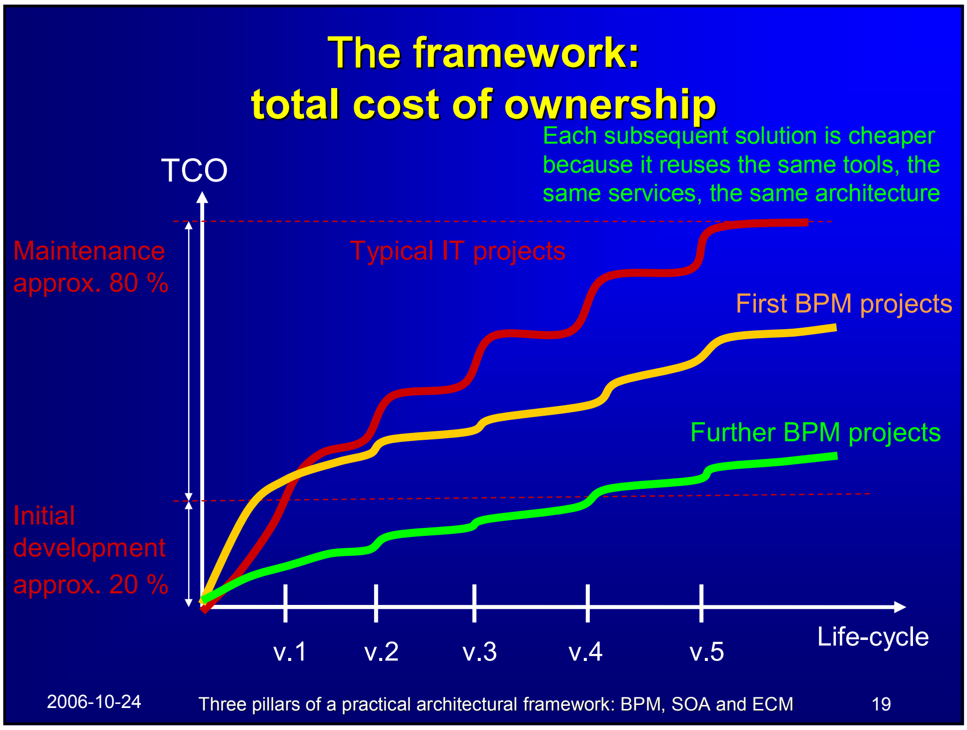
The framework: real agility achieved

- *Micro-projects* – agile implementations of new features
  - are carried out in a manner similar to Deming’s wheel
- *Meta-projects* – architectural framework governance for the management of many micro-projects
  - looks like maintenance rather than development



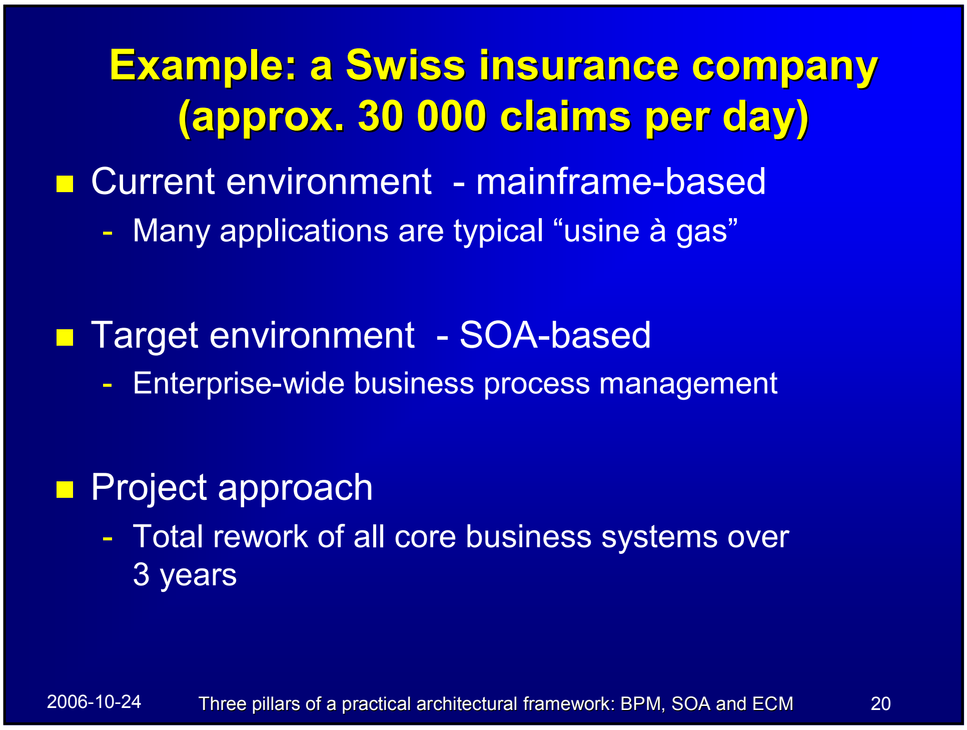
The framework: micro-projects life cycle

- Plan
  - fact- and rule-based selection of what should be done next as a micro-project
- Do
  - execution of a micro-project
- Check
  - new findings and solutions are considered for wider use
- Act
  - refactoring of the system



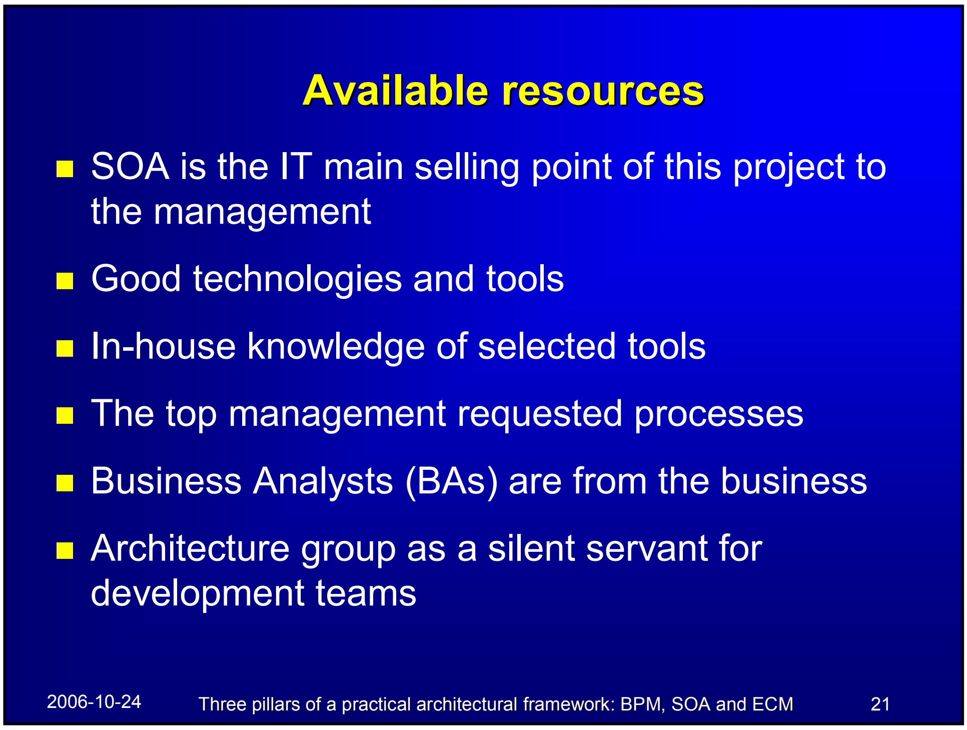
The framework: total cost of ownership

Each subsequent solution is cheaper because it reuses the same tools, the same services, the same architecture



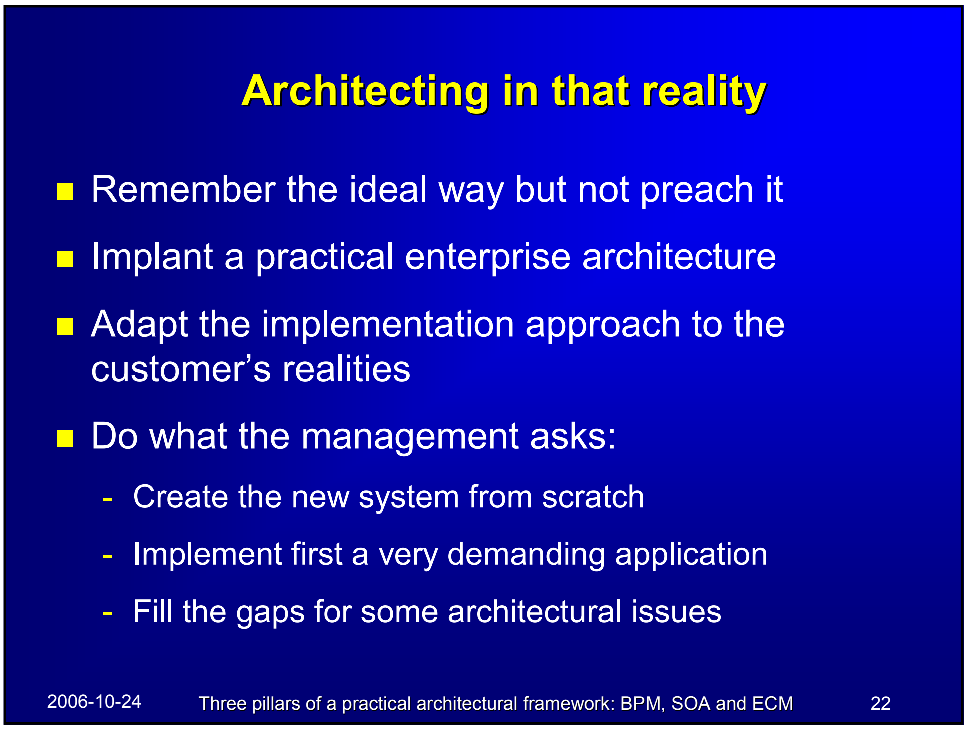
Example: a Swiss insurance company (approx. 30 000 claims per day)

- Current environment - mainframe-based
  - Many applications are typical “usine à gas”
- Target environment - SOA-based
  - Enterprise-wide business process management
- Project approach
  - Total rework of all core business systems over 3 years



Available resources

- SOA is the IT main selling point of this project to the management
- Good technologies and tools
- In-house knowledge of selected tools
- The top management requested processes
- Business Analysts (BAs) are from the business
- Architecture group as a silent servant for development teams



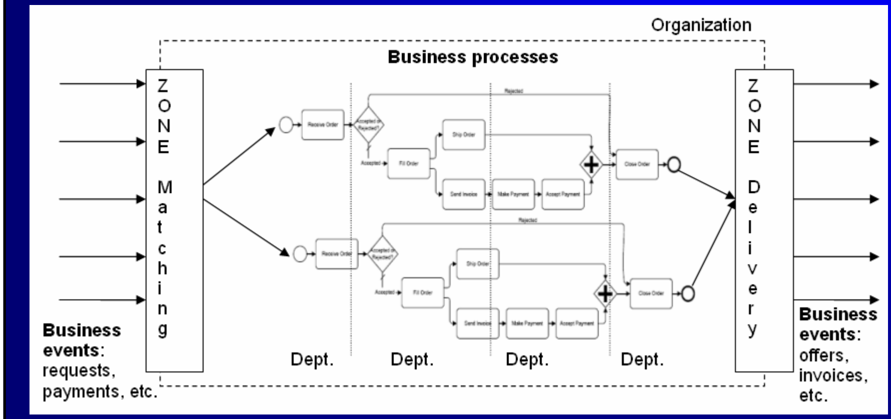
Architecting in that reality

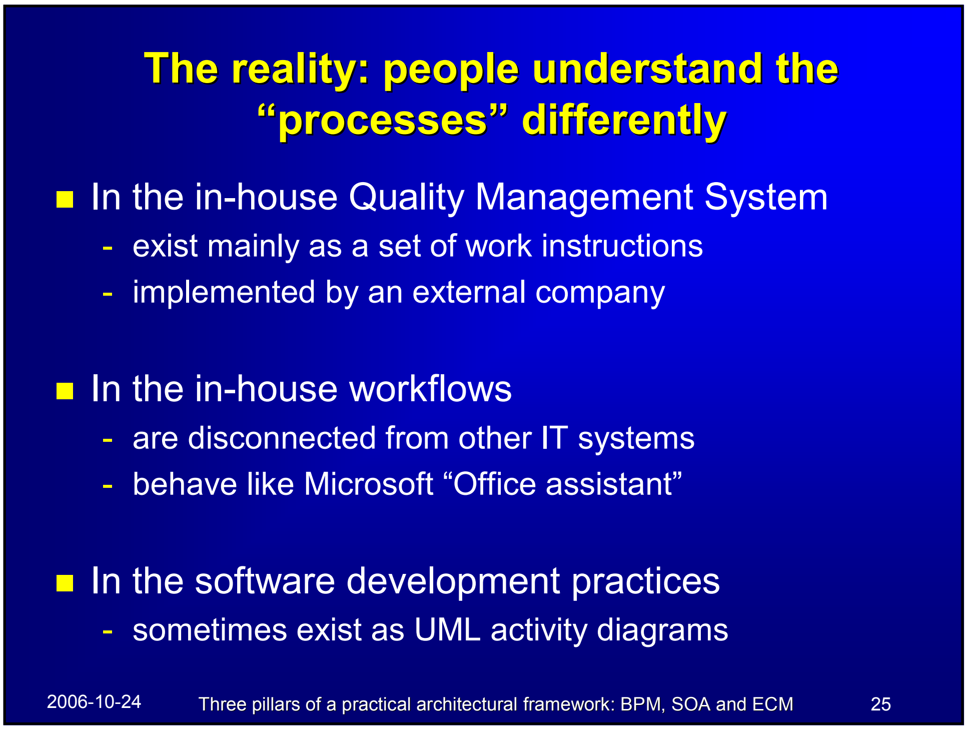
- Remember the ideal way but not preach it
- Implant a practical enterprise architecture
- Adapt the implementation approach to the customer’s realities
- Do what the management asks:
  - Create the new system from scratch
  - Implement first a very demanding application
  - Fill the gaps for some architectural issues

Common principles for design of the new system

- Big picture
- Business events
- Long-running processes
- Avoid dispersion of business logic
- Services, services, services
- Processes are first-class citizens
- Process modeling in Business Process Modeling Notation (BPMN)
- Process execution in Business Process Execution Language (BPEL)

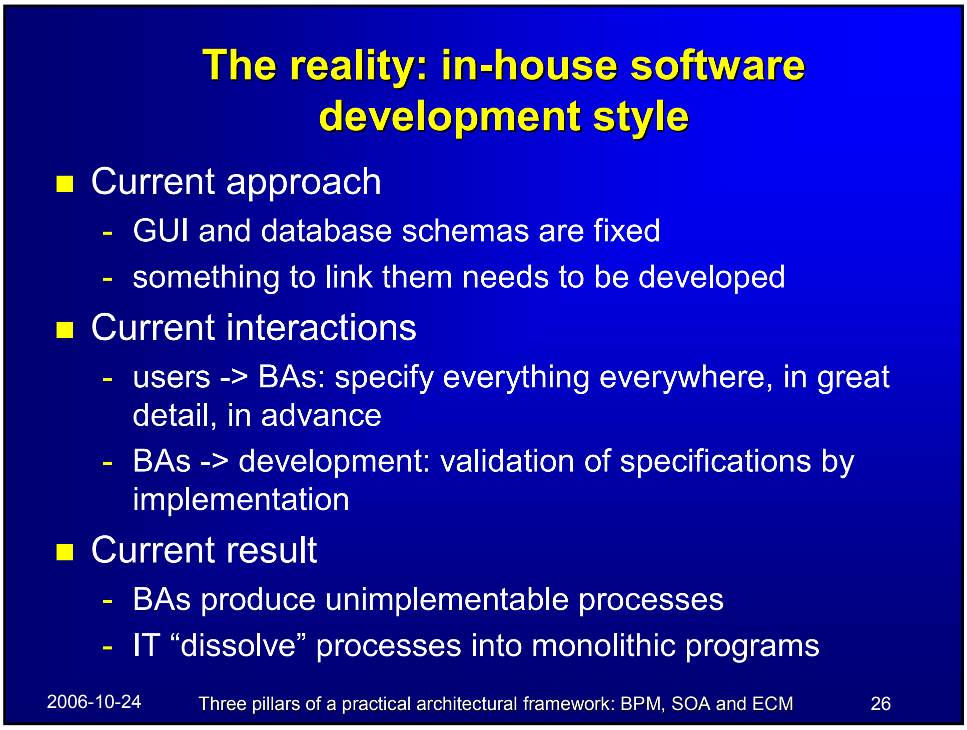
Big picture: typical service- and process-oriented enterprise





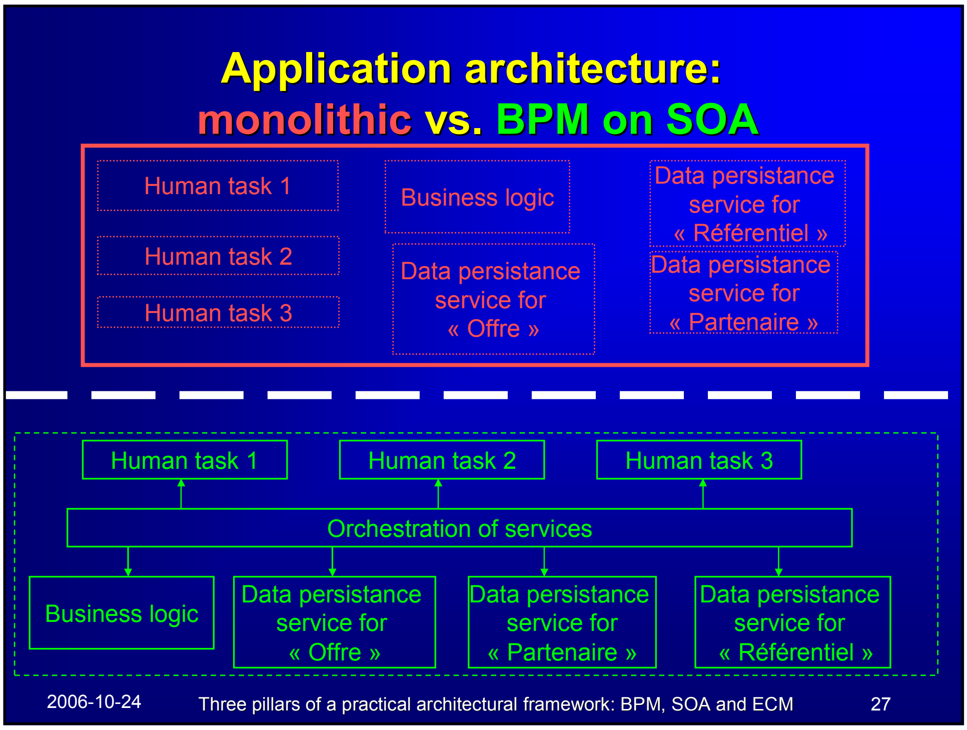
The reality: people understand the "processes" differently

- In the in-house Quality Management System
  - exist mainly as a set of work instructions
  - implemented by an external company
- In the in-house workflows
  - are disconnected from other IT systems
  - behave like Microsoft "Office assistant"
- In the software development practices
  - sometimes exist as UML activity diagrams



The reality: in-house software development style

- Current approach
  - GUI and database schemas are fixed
  - something to link them needs to be developed
- Current interactions
  - users -> BAs: specify everything everywhere, in great detail, in advance
  - BAs -> development: validation of specifications by implementation
- Current result
  - BAs produce unimplementable processes
  - IT "dissolve" processes into monolithic programs



Application architecture:
monolithic vs. BPM on SOA

Human task 1

Human task 2

Human task 3

Business logic

Data persistance service for « Offre »

Data persistance service for « Référentiel »

Data persistance service for « Partenaire »

Human task 1

Human task 2

Human task 3

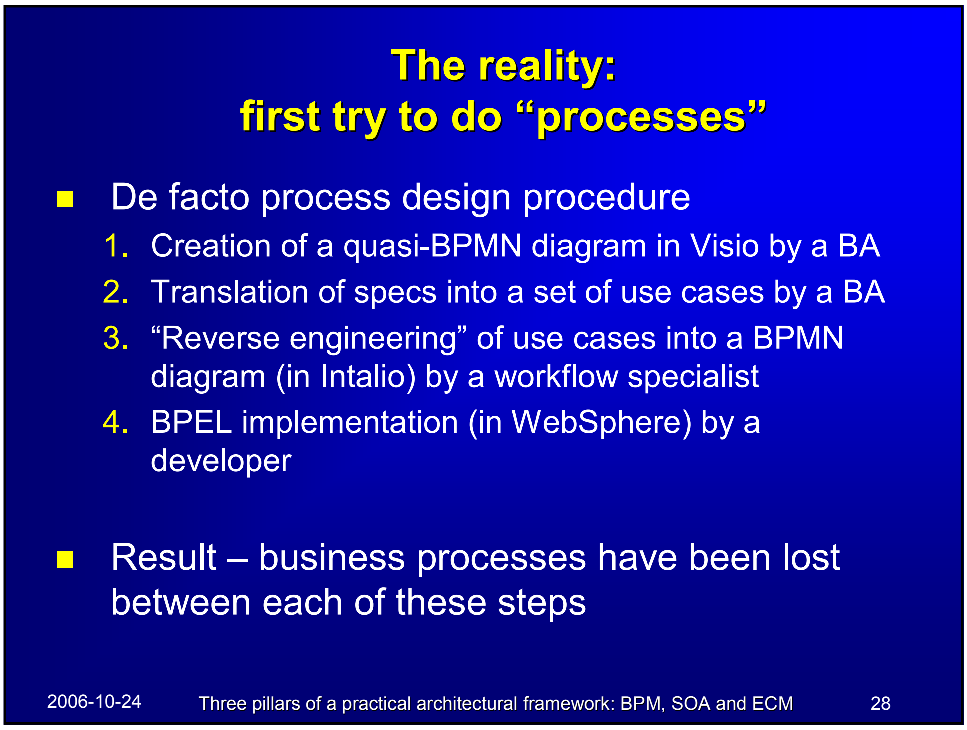
Orchestration of services

Business logic

Data persistance service for « Offre »

Data persistance service for « Partenaire »

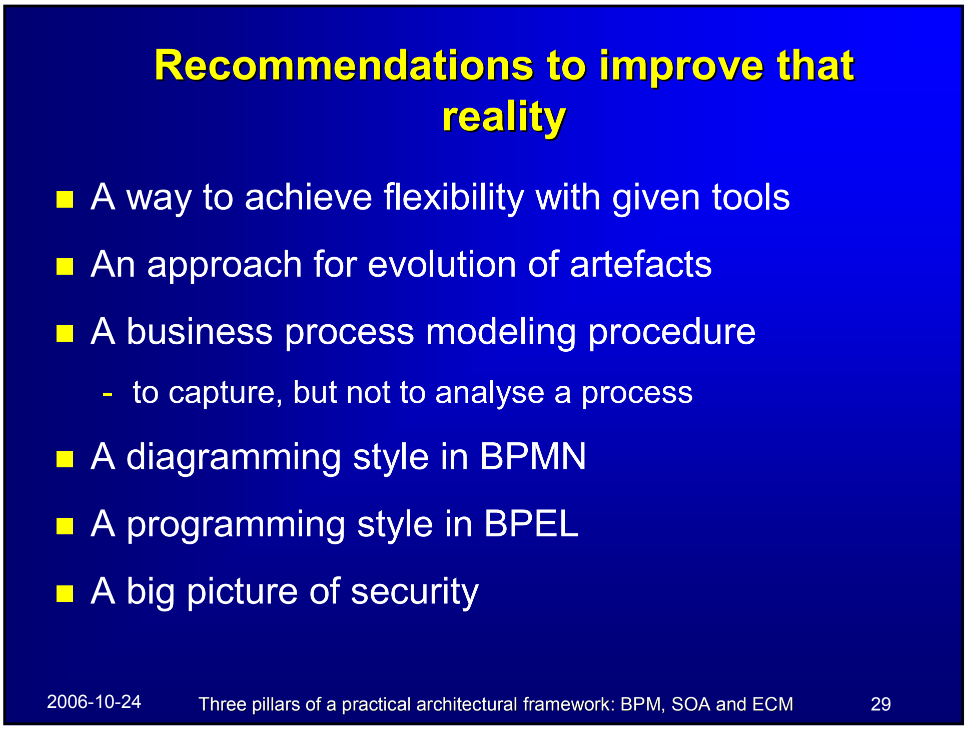
Data persistance service for « Référentiel »



The reality:
first try to do “processes”

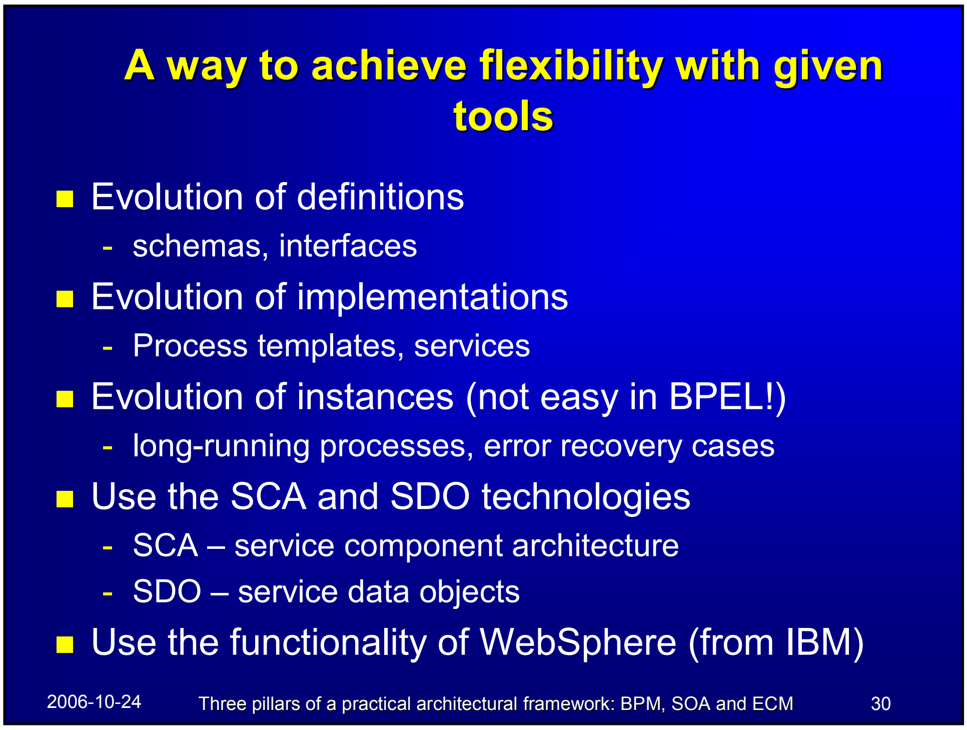
- De facto process design procedure
  1. Creation of a quasi-BPMN diagram in Visio by a BA
  2. Translation of specs into a set of use cases by a BA
  3. “Reverse engineering” of use cases into a BPMN diagram (in Intalio) by a workflow specialist
  4. BPEL implementation (in WebSphere) by a developer

- Result – business processes have been lost between each of these steps



Recommendations to improve that reality

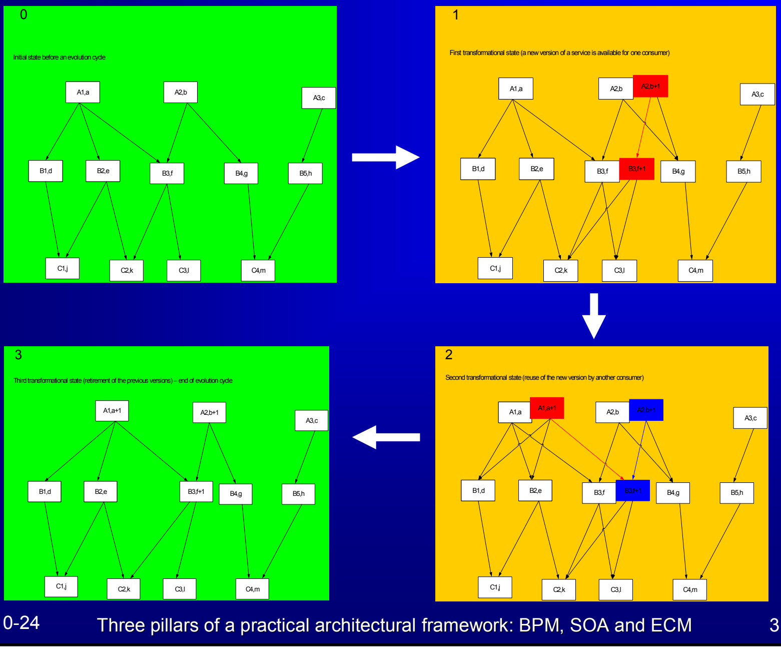
- A way to achieve flexibility with given tools
- An approach for evolution of artefacts
- A business process modeling procedure
  - to capture, but not to analyse a process
- A diagramming style in BPMN
- A programming style in BPEL
- A big picture of security



A way to achieve flexibility with given tools

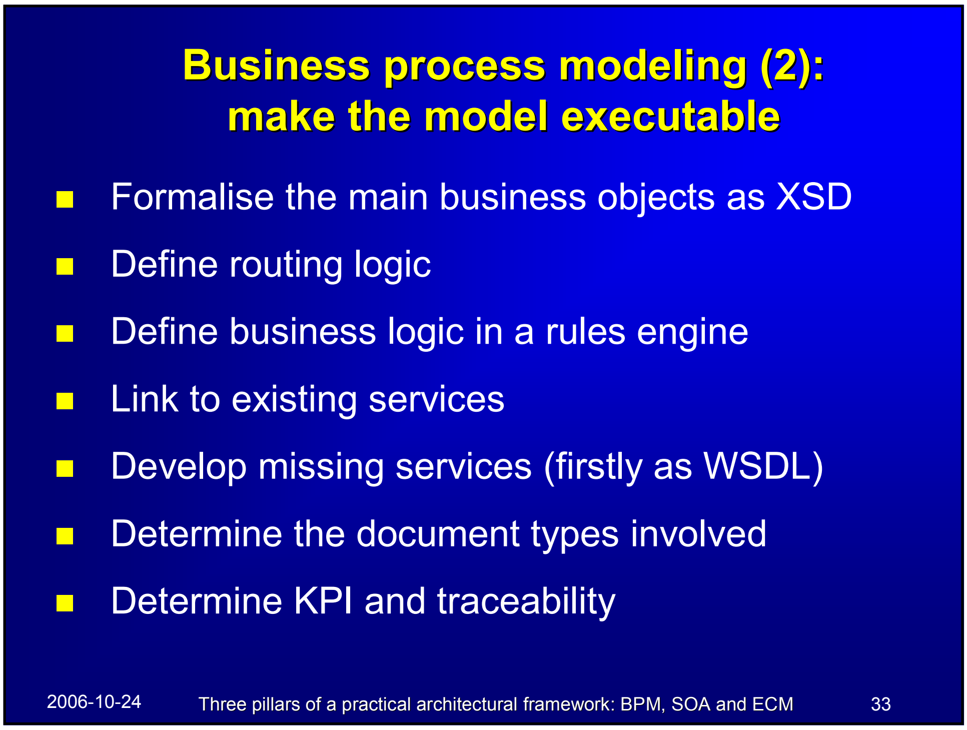
- Evolution of definitions
  - schemas, interfaces
- Evolution of implementations
  - Process templates, services
- Evolution of instances (not easy in BPEL!)
  - long-running processes, error recovery cases
- Use the SCA and SDO technologies
  - SCA – service component architecture
  - SDO – service data objects
- Use the functionality of WebSphere (from IBM)

An approach for evolution of artefacts


0
Initial state before an evolution cycle
A1,a
A2,b
A3,c
B1,d
B2,e
B3,f
B4,g
B5,h
C1,j
C2,k
C3,l
C4,m
1
First transformational state (a new version of a service is available for one consumer)
A1,a
A2,b
A2,b+1
A3,c
B1,d
B2,e
B3,f
B3,f+1
B4,g
B5,h
C1,j
C2,k
C3,l
C4,m
2
Second transformational state (reuse of the new version by another consumer)
A1,a
A1,a+1
A2,b
A2,b+1
A3,c
B1,d
B2,e
B3,f
B3,f+1
B4,g
B5,h
C1,j
C2,k
C3,l
C4,m
3
Third transformational state (retirement of the previous versions) – end of evolution cycle
A1,a+1
A2,b+1
A3,c
B1,d
B2,e
B3,f+1
B4,g
B5,h
C1,j
C2,k
C3,l
C4,m

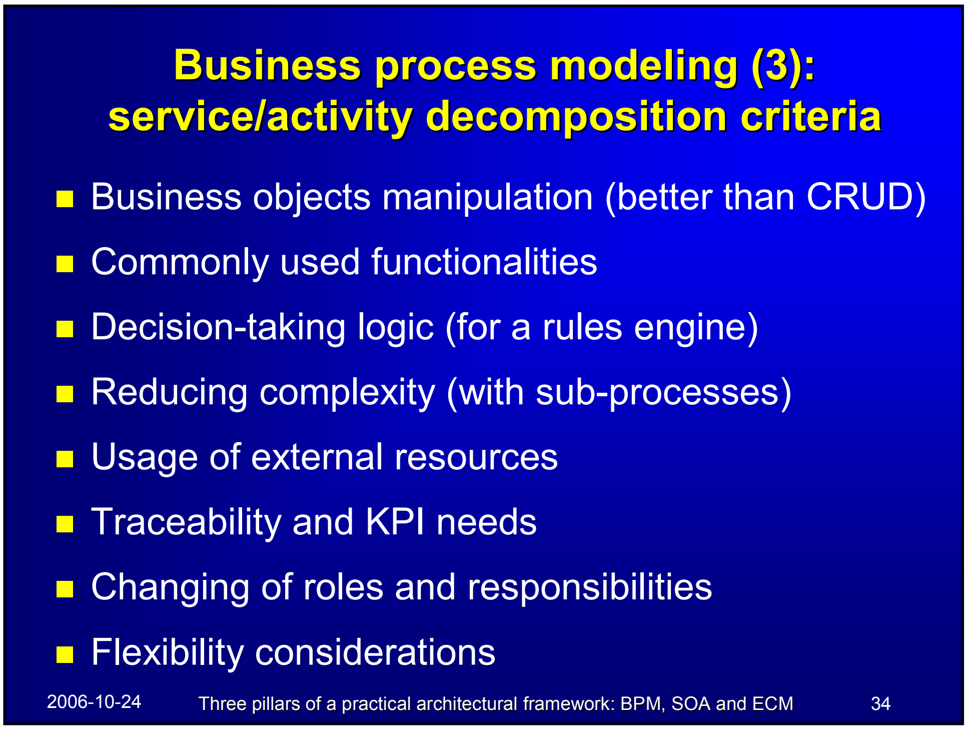
Business process modeling (1): create an aggregated model

1. Identify the main business objects
2. Determine related business events
3. List other business processes involved
4. Implement the flow of activities
5. Describe human activities
6. Describe automated activities
7. Document use cases



Business process modeling (2): make the model executable

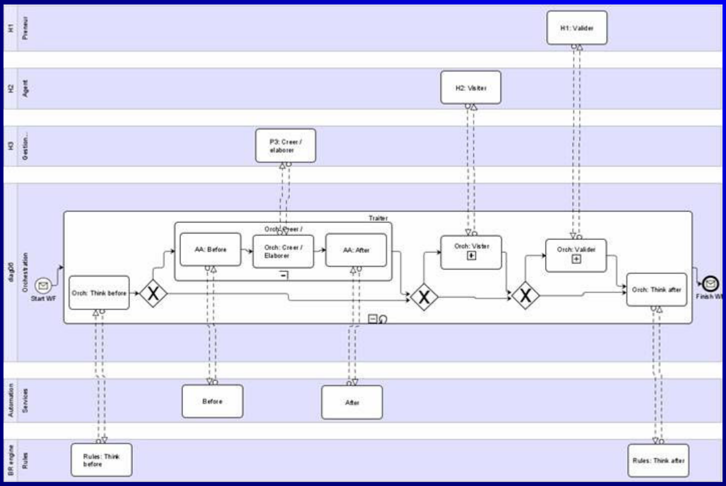
- Formalise the main business objects as XSD
- Define routing logic
- Define business logic in a rules engine
- Link to existing services
- Develop missing services (firstly as WSDL)
- Determine the document types involved
- Determine KPI and traceability



Business process modeling (3): service/activity decomposition criteria

- Business objects manipulation (better than CRUD)
- Commonly used functionalities
- Decision-taking logic (for a rules engine)
- Reducing complexity (with sub-processes)
- Usage of external resources
- Traceability and KPI needs
- Changing of roles and responsibilities
- Flexibility considerations

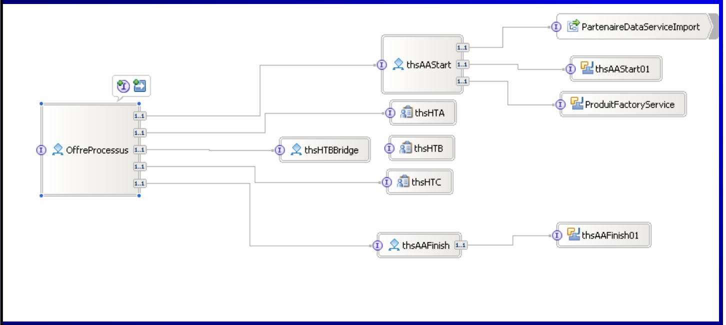
A diagramming style in BPMN

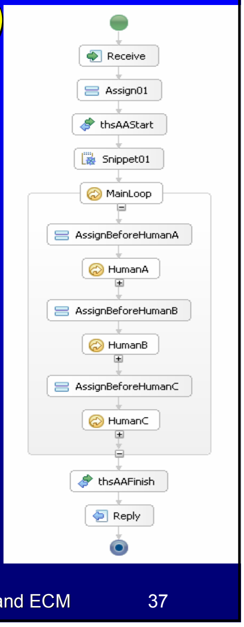


A common tool for business and IT: Designer from www.intalio.com

- BPMN diagrams serve for validation of specs
  - Artefacts defined by BAs are quickly implemented
  - Processes are always executable
  - Possibility to use existing services
  - Simple automated testing is reflected
- Encourage BAs to use Designer directly
- Discuss BPMN diagrams with the users

A programming style in BPEL (with SCA in WebSphere)





Conclusions

- Tools and technologies are available
- Enterprise architecture is affordable
- Use it to architect future flexibility
- Implementation matters
- The use of common tools by both the business side and the IT side is of great benefit
- Neither technology nor any architecture works if the politics don’t fly

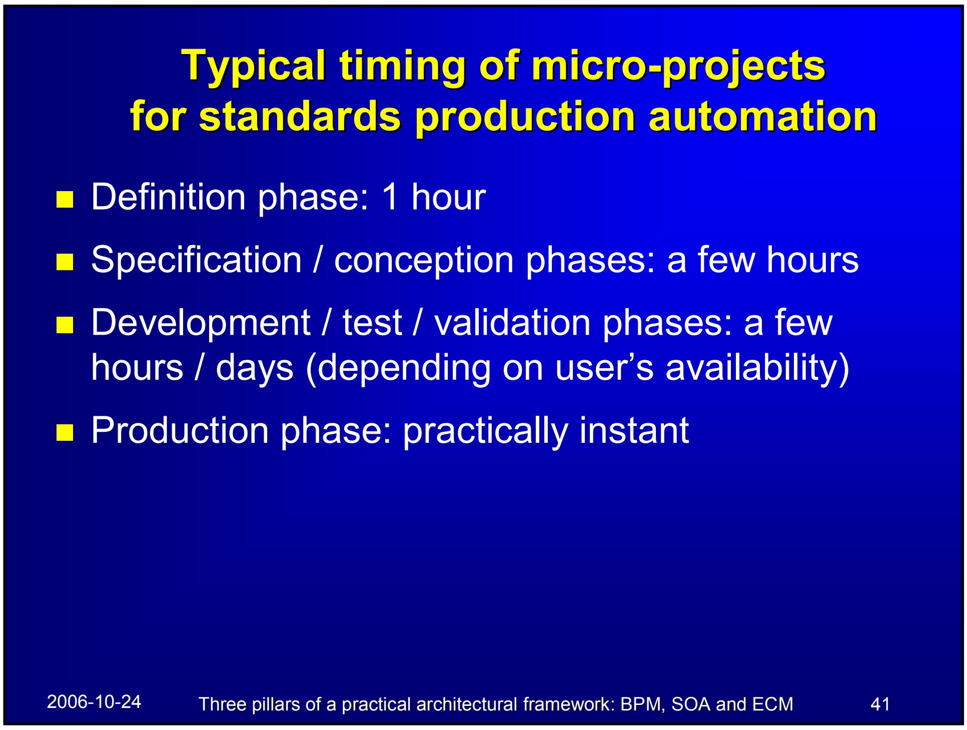
THANK YOU

- Questions and answers

- samarin@bluemail.ch
- www.clio.ch
- +41 76 573 40 61

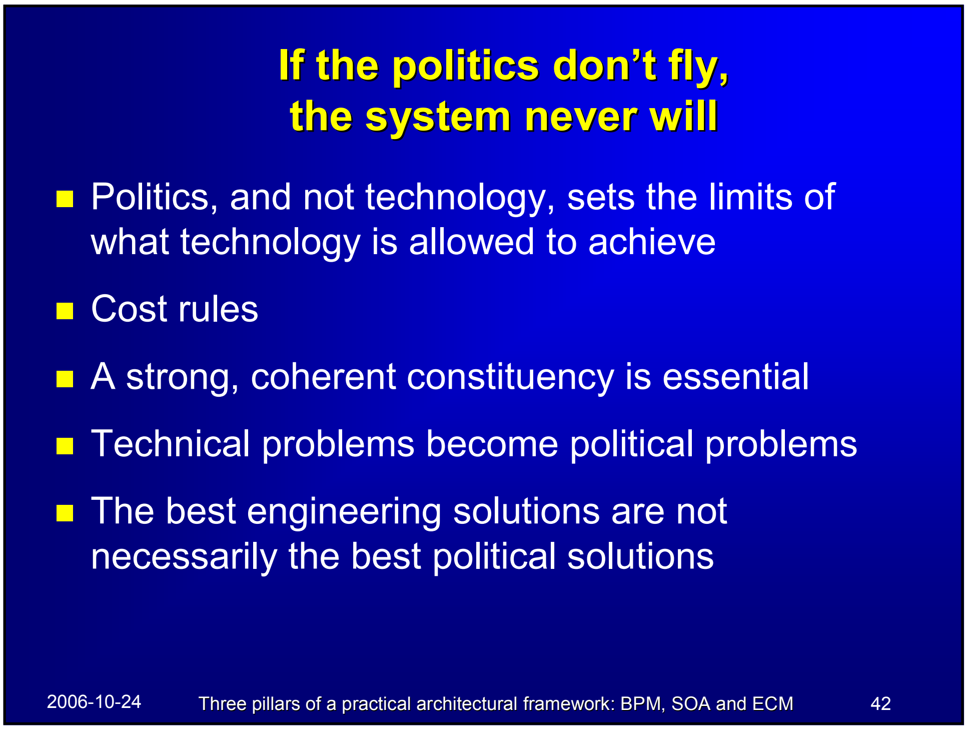
Description

- The aim of this talk is to share my experience in the use of a practical architectural framework for the improvement of complex business systems. This framework is a generic enterprise architecture which
  - provides a systemic approach, adaptability and flexibility,
  - brings together people, processes and programs,
  - deals with business events, procedures, rules, objects and data,
  - allows agile step-by-step deployment — each step is a mini-project carried out at the users’ pace,
  - economises considerably the required resources, and
  - is designed for transforming a historically-grown complex system of systems into a coherent, smoothly evolving, business environment.



Typical timing of micro-projects for standards production automation

- Definition phase: 1 hour
- Specification / conception phases: a few hours
- Development / test / validation phases: a few hours / days (depending on user's availability)
- Production phase: practically instant



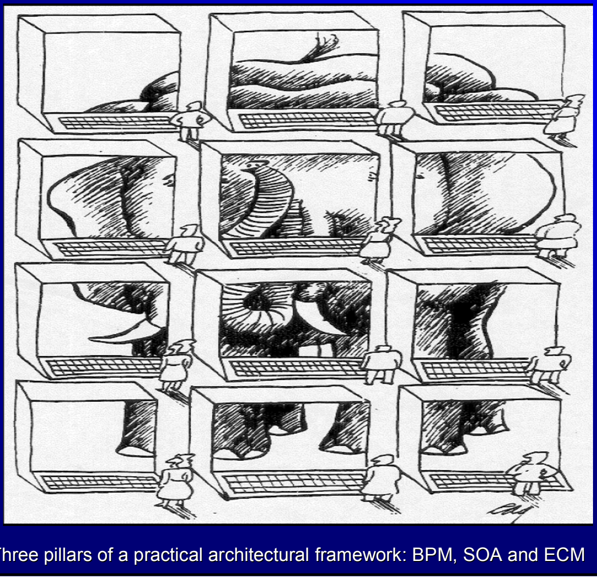
If the politics don't fly, the system never will

- Politics, and not technology, sets the limits of what technology is allowed to achieve
- Cost rules
- A strong, coherent constituency is essential
- Technical problems become political problems
- The best engineering solutions are not necessarily the best political solutions

My position: agile means adaptable

- Agility is the ability not only to **create change** but to **respond to change**
- Agility is the ability to balance flexibility and structure
- Gartner: Agility is the ability of an organization to sense environmental change and to respond to it efficiently and effectively

The main lesson from agile development: see and understand the big picture



Management of micro-projects

- In-depth knowledge of the domain is essential
- Sharing "the vision" with the business process owner
- Architecting the product (i.e. a business process) in terms of IT and business systems
- Guiding the project team to implement the product
- Giving practical help, if necessary
- Facilitate, influence and coordinate rather than control and act as (or consider yourself to be!) the ultimate authority