From agile development to agile evolution of enterprise systems

Dr Alexander Samarin

www.samarin.biz

Clio S.A.

EuroPython conference 2006,
CERN, Geneva, Switzerland

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 academic, 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")

Agile software development is a classic case of disruptive technology

- The customers appreciate it
  - an order of magnitude increase in IT value for the customers
  - no comprehensive up-front specs
  - results-oriented

- The IT establishment criticises it
  - bad or no design
  - no documentation
  - works only for top professionals

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

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 flexible business system which is easily adaptable to:
  - policies, priorities, existing data, IT systems, business processes, size, complexity, budgets, culture, etc.
- Subject to socio-technical aspects:
  - **how** you do something may be more important than **what** you do

Lean production is an example of optimisation in industry

- See the whole picture
- Learn constantly
- Decide as late as possible
- Deliver as fast as possible
- Eliminate waste
- Empower the team
- Build in integrity
- Avoid sub-optimisation

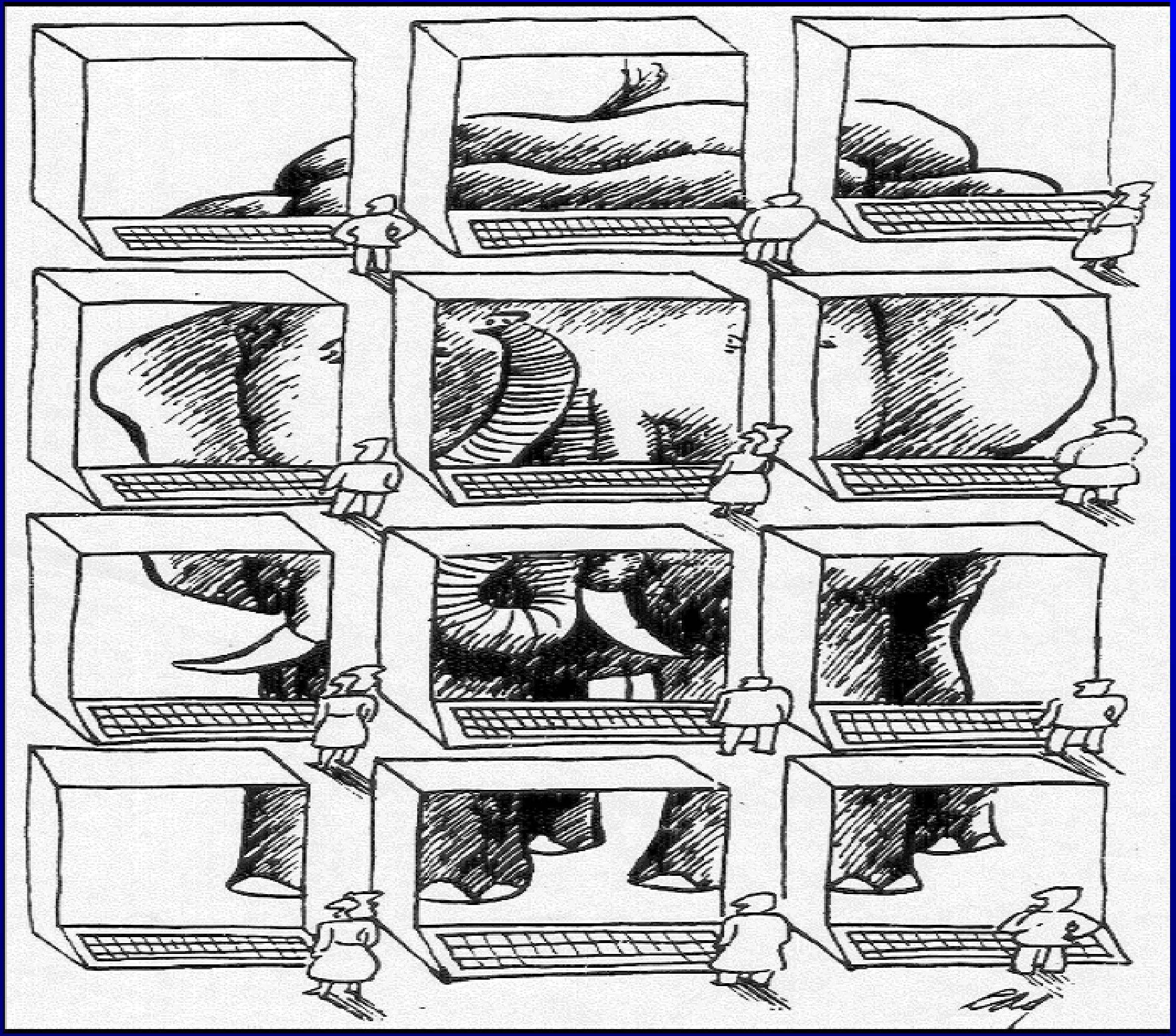
Harvard Business School studies: development practices that spell success

- Early release of the evolving product design to customers
- Daily incorporation of new software code and rapid feedback on design changes
- Teams with broad-based experience of shipping multiple projects
- Major investments in the design of the product architecture

A dilemma

- Agile software development is sound
- … although there are valid criticisms (agile development is like racing a yacht and is not necessarily suited to captaining a liner)
- We need it to deliver agile evolution of enterprise systems
- Heuristic: sometimes it is necessary to **expand the *concept*** in order to simplify the problem

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



Critical aspects for agile evolution of enterprise systems

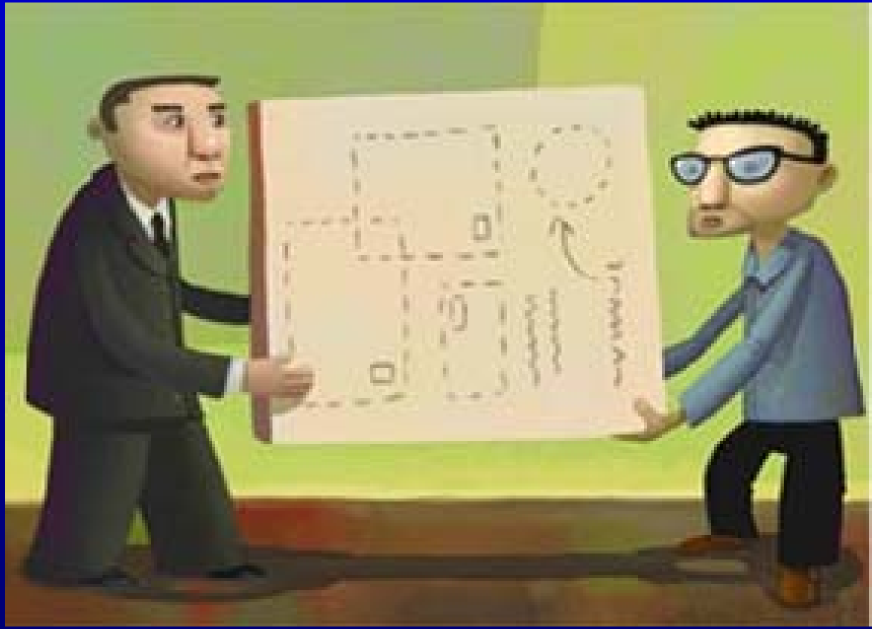
- The big picture (i.e. the enterprise architecture) is
  - available
  - understood
  - agreed internally by consensus
  - not “parachuted” by consultants or a vendor
  - addressing Business Process Management (BPM)
- Development of the architecture for a particular enterprise should not be a project that takes many man-years and reams of pages
- Business and IT use common tools

Enterprise means Business Process Management

- Business Process Management (BPM) allows you to model, automate, control, measure and optimise the flow of business process steps
- It spans your organisation’s systems, people, customers and partners within and beyond your corporate boundaries
- Gartner estimates that there are currently over 140 business process management vendors

My approach to BPM (providing a fishing rod, not a fish)

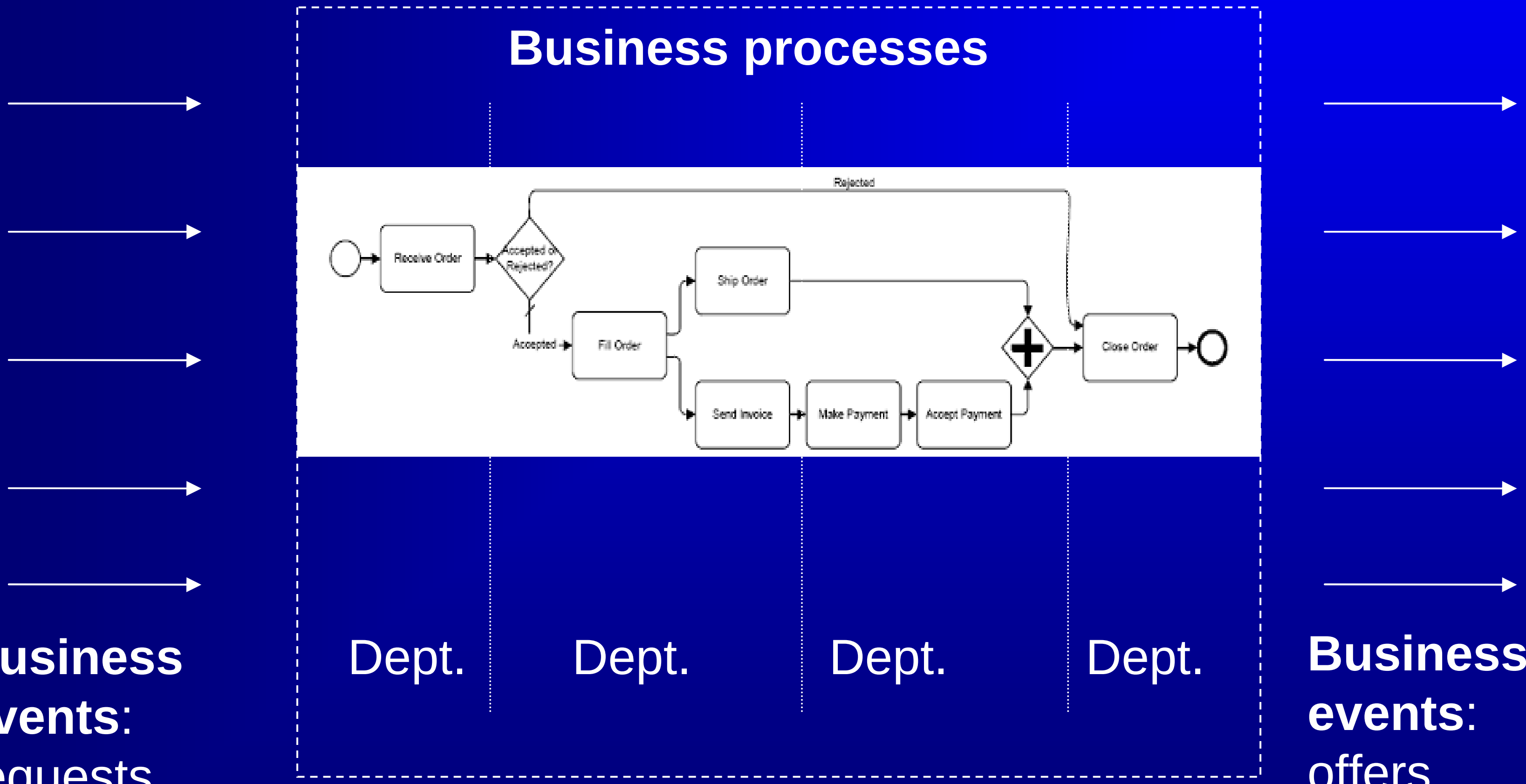
- Any enterprise has its own BPM system; some enterprises want to change it
- An offer of an architectural framework for improving BPM systems
- It makes use of the synergy that exists between business needs and IT potentials
- Designed for agile evolution



Main characteristics of the architectural framework

- Systemic approach and adaptability
- Generic operational model (for business)
- Advanced multi-layer model (for IT)
- New features are added like pieces of Lego
- Proven that BPM systems can be improved faster, better and less expensively

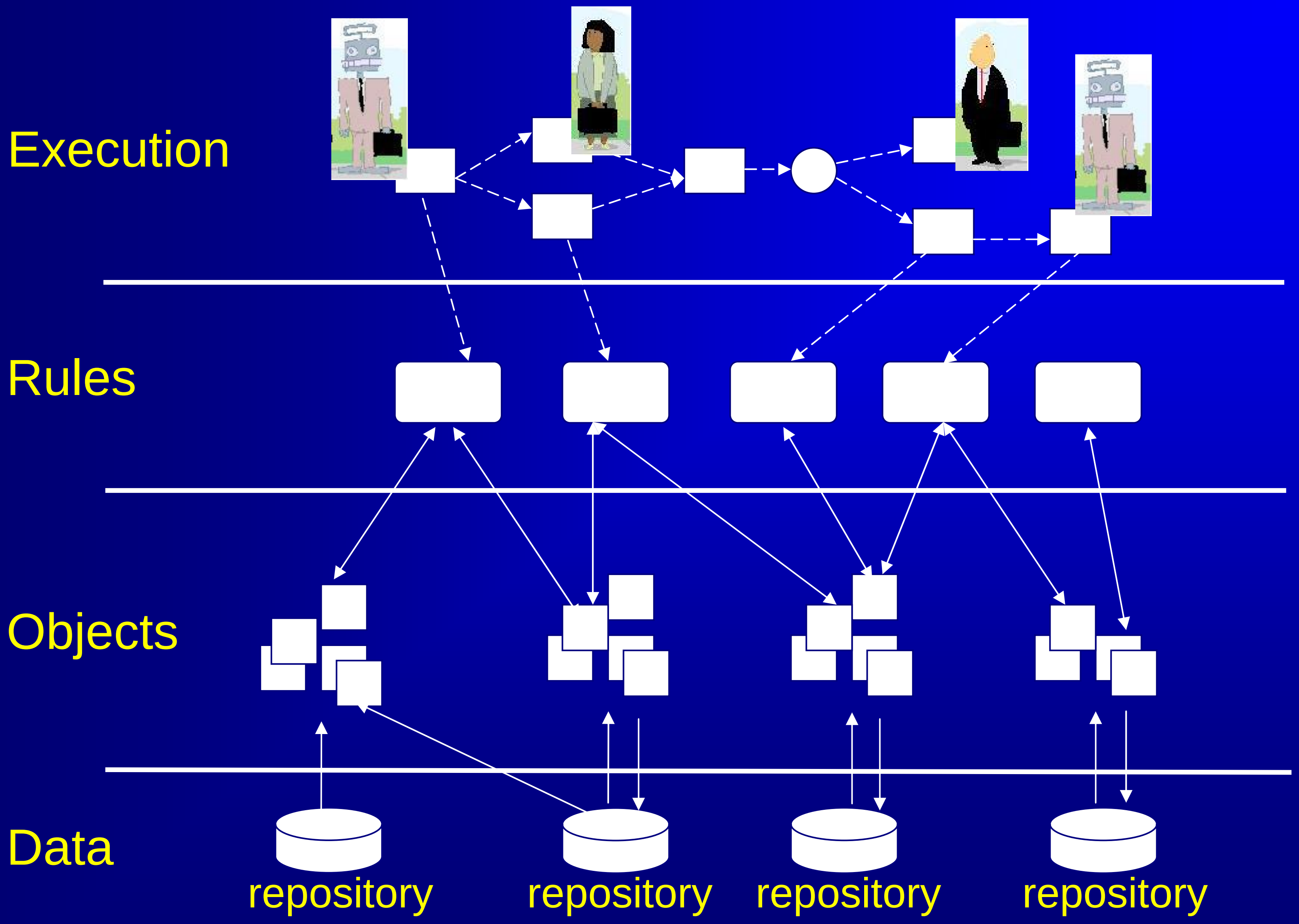
Typical service and process oriented enterprise



Business events: requests, payments, etc.

Business events: offers, invoices, etc.

The simplified multi-layer model

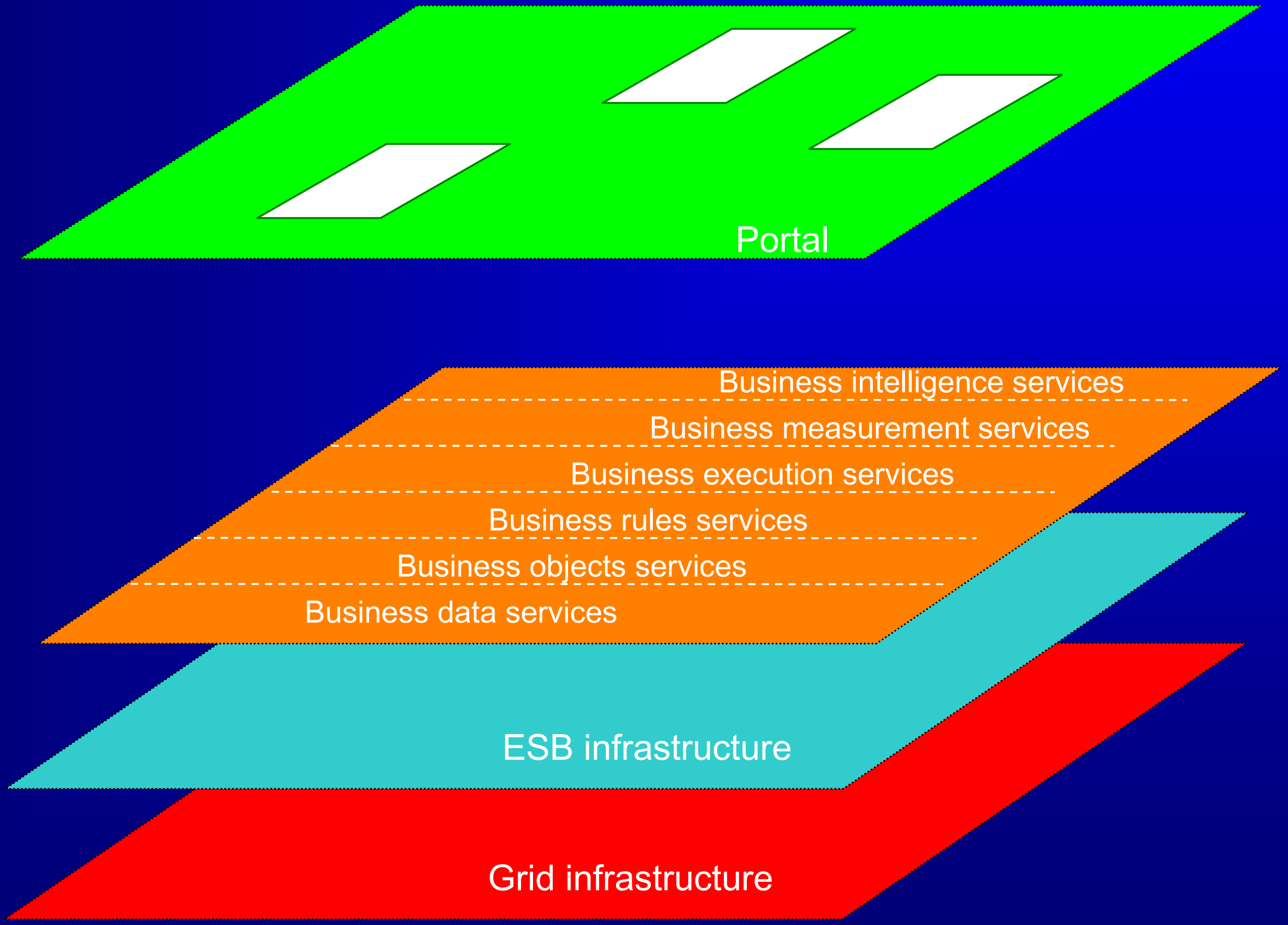


Why this approach produces agile systems

- Many of the difficult issues are resolved:
  - Architecture, Methodology, Patterns
- There *is* no “classic” application – instead, there is a set of orchestrated services
- Services those are versionable and clonable
- 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

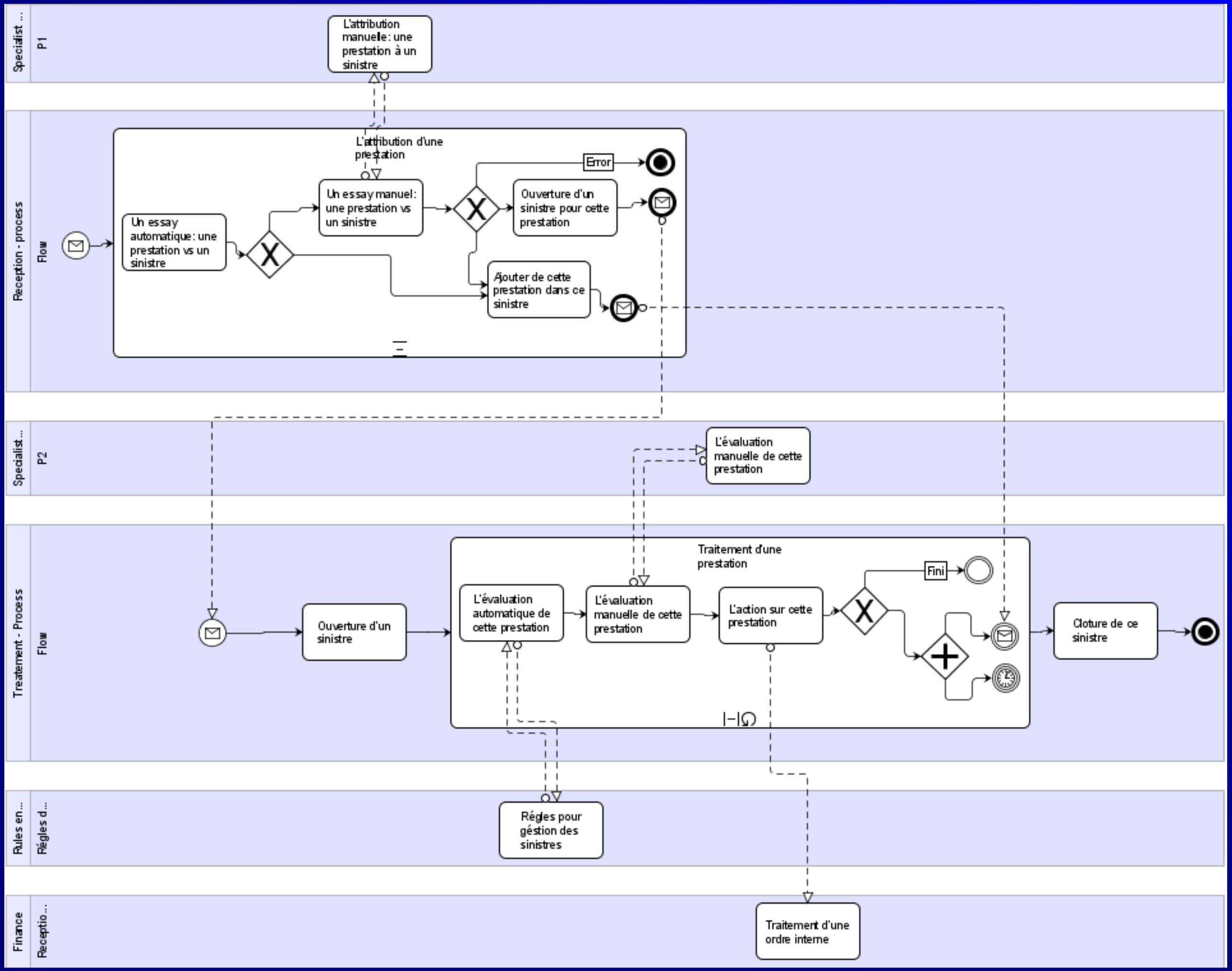
Works with other technologies


Portal
Business intelligence services
Business measurement services
Business execution services
Business rules services
Business objects services
Business data services
ESB infrastructure
Grid infrastructure

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) and they are wrapped into services
- It generally does not have its "own" database
- It is "outside" existing systems
- It reuses existing services

A common tool with BPMN and BPEL (e.g. www.intalio.com)



Specialist ...
P1
L'attribution manuelle: une prestation à un sinistre
Reception - process
Flow
L'attribution d'une prestation
Un essay automatique: une prestation vs un sinistre
Un essay manuel: une prestation vs un sinistre
Error
Ouverture d'un sinistre pour cette prestation
Ajouter de cette prestation dans ce sinistre
Specialist ...
P2
L'évaluation manuelle de cette prestation
Treatement - Process
Flow
Ouverture d'un sinistre
Traitement d'une prestation
L'évaluation automatique de cette prestation
L'évaluation manuelle de cette prestation
L'action sur cette prestation
Fini
Cloture de ce sinistre
Rules en...
Règles d...
Régles pour géstion des sinistres
Finance
Receptio...
Traitement d'une ordre interne

Many thanks to Jython

- Excellent as the glue between enterprise applications
- Easy to manage (e.g. fragments are kept in .jar)
- Highly flexible (i.e. introspection)
- Dynamic loading and execution
- The only thing that was added:
  - a .py wrapper to simplify execution

Real agility achieved: two types of project

- *Micro-projects* – agile implementations of new features

- *Meta-projects* – architectural framework governance for the management of many micro-projects
  - looks like maintenance rather than development

Meta-projects are carried out in a manner similar to Deming’s wheel

- 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

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 the ultimate authority

Micro-projects: definition phase

- Business optimisations are evaluated
- Features to be implemented are understood
- Priorities (availability of features) are communicated

Micro-projects: specification / conception phases

- A prototype of a product (e.g. an executable business process diagram) is used to validate what will be implemented
- Missing components (if any) are identified and specified
- Missing components (if any) are evaluated; use of new tools/utilities should be justified

Micro-projects: development / test / validation phases

- Missing components (if any) are incrementally developed, tested and validated
- Whole product is assembled from available and new (started as dummies) components
- Whole product is incrementally tested, validated and deployed
- Extra monitoring (if necessary) is deployed
- Necessary resources are estimated and the system is reconfigured to provide them

Micro-projects: production phase

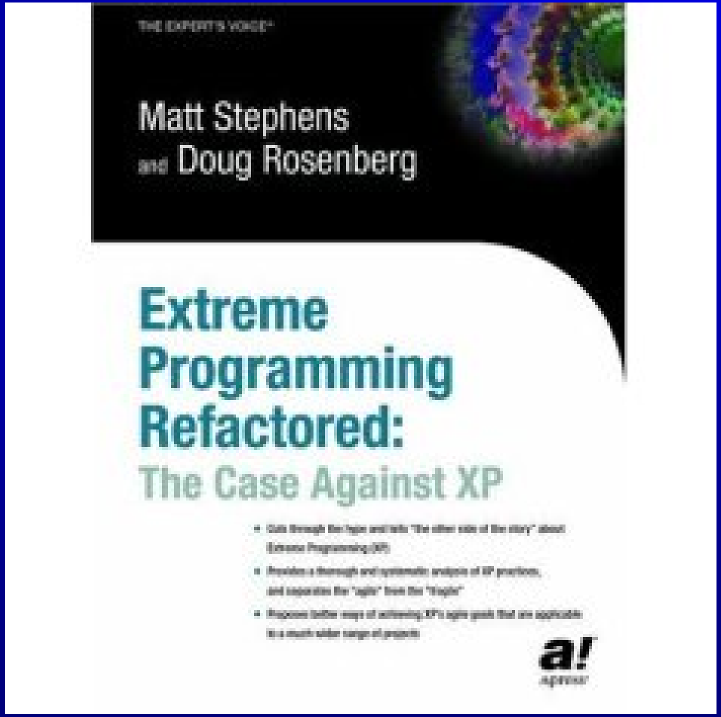
- New product or new version of a product is declared available for use by the business or by other components
- In the case of replacement, some estimations are made when the previous version of a product is to be discontinued

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

Address some criticisms from the following book

- "Extreme Programming Refactored: The case against XP"



THE EXPERT'S VOICE®

Matt Stephens and Doug Rosenberg

Extreme Programming Refactored: The Case Against XP

- Cuts through the hype and tells "the other side of the story" about Extreme Programming (XP)
- Provides a thorough and systematic analysis of XP practices, and separates the "agile" from the "fragile"
- Proposes better ways of achieving XP's agile goals that are applicable to a much wider range of projects

a! Apress®

“Extreme culture”
(jumping straight to code)

- There are no obstacles
  - General design and patterns are available
  - Use a common tool for business process mapping
  - Can start with the existing process and refine it
  - A solution is firstly and quickly coded in executable business process diagramming language

“The on-site customer” (as a replacement of requirements)

- Close work with the customer is mandatory
  - It is just about changing the IT attitude towards the users
  - From master to service provider
  - From talking 95 % about IT issues and 5 % about business issues to a 50 % - 50 % balance
- Follow one of the principles of the Toyota Production System (TPS):
  - Go and see for yourself to understand thoroughly the situation

“Pair programming” (for compensating absence of design, documentation, etc.)

- From permanent pair programming to systematic code reviewing
  - Code review is necessary to keep conceptual integrity
  - Programming is a way for communication between humans
  - It is a good way to involve junior programmers

- Follow one of the principles of the TPS:
  - Grow leaders who thoroughly understand the work, live the philosophy and teach it to others

“Oral documentation” (documentation in XP tends to be paradoxical and confusing)

- A lot of documentation is already available, but often IT rejects it:
  - Architectural design document is too complex
  - Workflow diagrams are too much business
  - Jython code requires some formal training
  - Quality documentation (considered bureaucracy)

“Constant refactoring after programming” (If it ain’t broken, fix it anyway)

- Any enterprise system is never finished
  - Business and IT meaning of “broken” are rather different
  - Features are added if needed
  - It is a very dynamic system

- Follow one of the principles of the TPS:
  - Become a learning organisation through relentless reflection and continual improvement

Lessons learnt

- Combining the architectural framework and agile software development allows building of agile enterprise systems
- The use of common tools by both the business side and the IT side is of great benefit
- Neither agile development nor any architecture works well if the politics don’t fly

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

THANK YOU

- Questions and answers

Short description

- The aim of this talk is to share my experience in the use of a practical architectural framework for the improvement of enterprise business systems. This framework uses a dynamic language (i.e. Jython) and agile development practices. Experience with business acceptance and composition of applications are discussed.
- This presentation is complementary to my presentation at Plone conference 2005 in Vienna (“The use of Plone for enterprise solutions”) – it is focused on the technical and practical aspects of agile evolution of enterprise

Architectural framework experience (1)

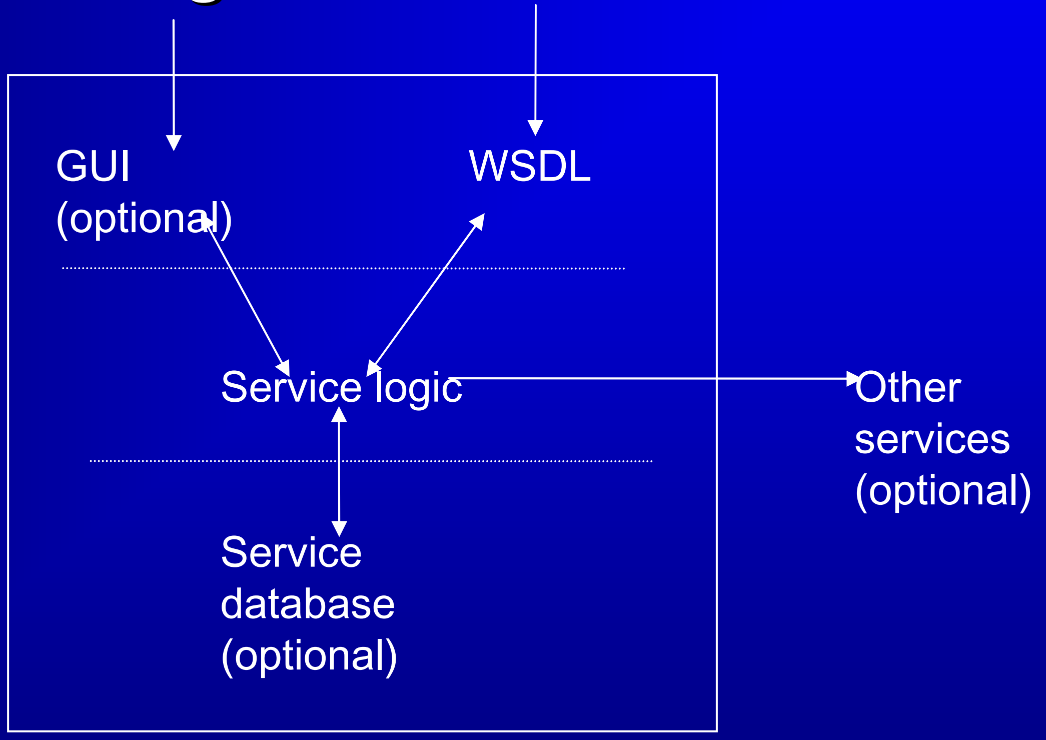
- Architectural framework provides a basis for technical and political decisions
- IT development is unified
- The target is business process automation and not applications development (brings greater flexibility)
- The system is built mainly from commodity functional blocks (commercial and freely available) that are not modified

Architectural framework experience (2)

- The system is built incrementally (quicker ROI)
- High level of user involvement
- Team has ownership and customizes its approach
- Project retrospection (no post-mortem or witch hunt)
- Achieve synergy between practices, principles and experiences for building software systems
- Take advantage of the organisational business and organisational knowledge

Based on SOA:

In general, a layer, a building block and a version of a building block are all services



Micro-projects: documentation

- Built-in quality management system
- Use of visual tools (e.g. business process diagrams, forms)
- User management, maintenance and programming
- IT-specific documentation is mainly available from the meta-project and adopted patterns
- Programs as documents and documents as programs