Smalltalk in Large-Scale Enterprise Architectures

Rob Vens

http://www.sepher.com

rob@sepher.com

Traditional Smalltalk Applications

- Client-Server
- Fat client
- GUI intensive

Recent developments

- Web enabled
- Using web standards
  - HTML
  - XML
  - Soap
  - WDSL
  - UDDI
- Poor support for distribution, connectivity (i.e. with Java and J2EE)

Smalltalk market share

- Negligible – (don't blame ESUG)

- Largest in 1994 (according to STIC) when Smalltalk was competing with C++
- Steep decline since 1995 when Sun announced Java
- Now a niche player?

Can you sell Smalltalk to your management?

- Proven technology is what they want
- If it's not Java it's not modern
- Or the old arguments:
  - Smalltalk is slow
  - Too pure OO
  - Object-orientation has failed

The Java onslaught

- Many (if not most) Smalltalk developers moved to the Java world
- Many see Microsoft .NET as a more attractive alternative, with possibilities to continue to work with Smalltalk (Dave Simmons' *SmallScript)*

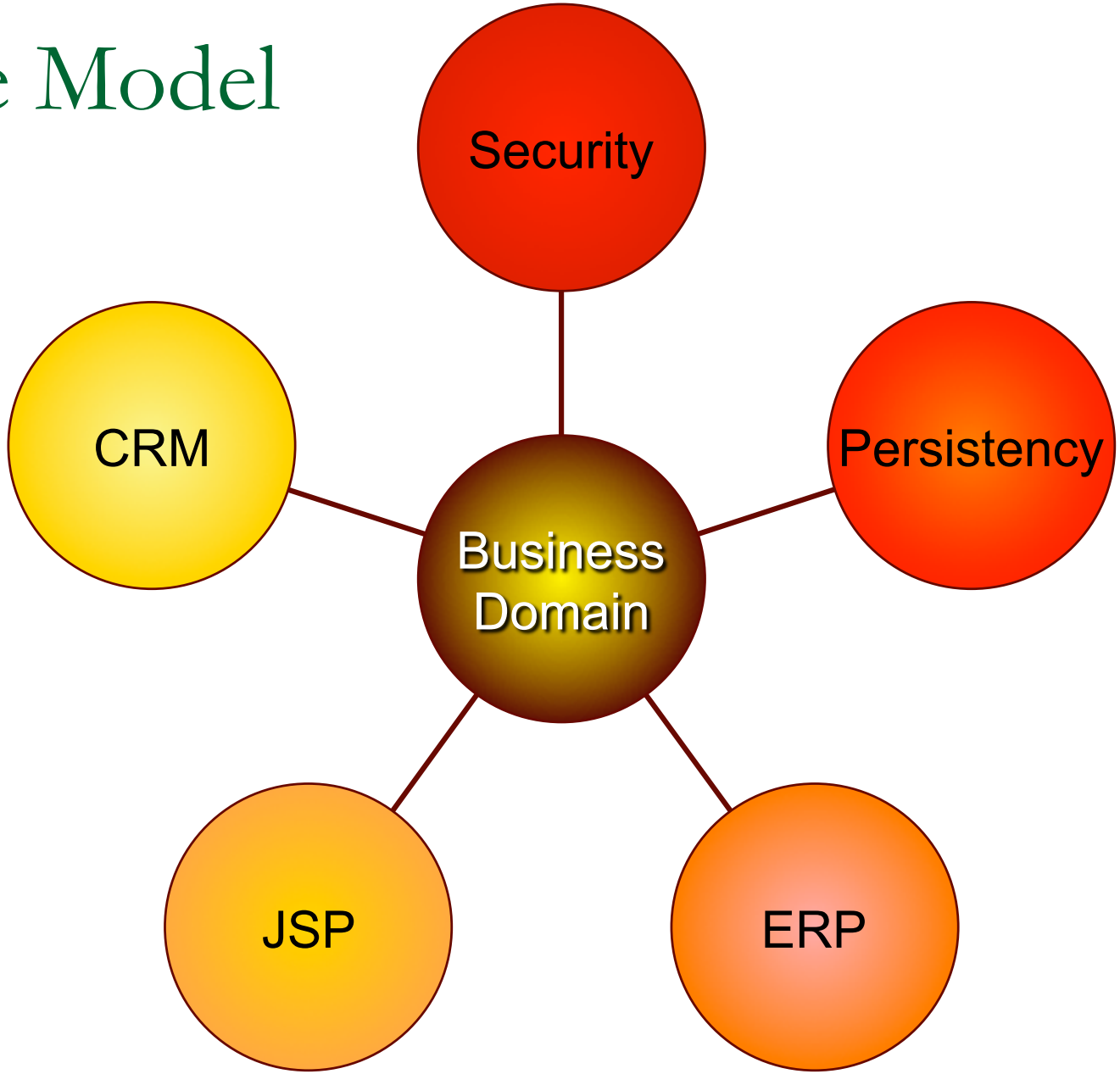
But there is another alternative …

- Basic idea is to use Smalltalk as an Enterprise Application Integrator (EAI)
- Several architectures are possible to do this
- I will propose a business-centred architecture

What is a business-centred architecture?

- Hub-and-spoke architecture
- All business logic in the hub
- Publish-and-subscribe mechanism in the spokes
- Adapters implementing the spokes
- Smalltalk in the hub, anything else in the rest

Core Model



What is business-centred?

- All business logic is concentrated in one logical component
- There is no business logic in any other component
  - Esp. ERP, CRM
  - Also messaging middleware is connected without business logic in the middleware tier
- This component is placed in the hub

Why Smalltalk?

- Smalltalk is eminently suited for the business logic component because:
  - Language and problem domain are closer than any language I know
  - For the business domain component other concerns are important (vs. service components):
    - Flexibility
    - Extensibility

Characteristics of the domain component

1. There is an OO model of the business
2. This model is a Roger Rabbit model
3. The model is written in UML
4. It is implemented in Smalltalk as an executable
5. It attempts to be an exact replica of the business in software – a kind of simulation model

OO model of the business

- Not new for many of you
- Recapitulating:
  - OO is (as far as I know) the only modelling tool that effectively deals with complexity
  - Only OO models can deal with the scalability problem (Alan Kays dog house metaphor)
  - Three alternatives exist:
    - Process modelling
    - Data modelling
    - Distributed agent models

Roger Rabbit models

- Also called: “active objects”
- What is active in the “real” world is passive in the model and vice-versa
- Business processes unfold in a backward-chaining process of objects delegating responsibilities (Responsibility Driven Design, CRC sessions)
- Process model is a “pull model”
- “Out of Control”

UML

- Smalltalk can be the modelling language but I hope we can agree that this is not ideal
- UML models need to be executable (OMG target in 2.0 and MDA)
- Close mapping between programming language and UML needed for the business component
  - UML support needed in IDE's!!!

The Smalltalk executable

- Logical component:
  - Can be implemented distributed
    - EasyBoard model
    - CORBA
    - Others …
  - Probably needs fault-tolerance support (question for the audience)
- Contains no technical issues (i.e. database transparency, user interface unaware, etc.)

Modelling issues: Simulation science

- The running executable is like a running simulation
- Executable models need to deal with dynamic behaviour, esp.:
  - Waiting lines
  - Stochastics
- Smalltalk has deep roots in simulation!

Criterion

The business component

can and will run

with all other components unavailable

Hub-and-spoke: the spokes

- This is where Java (or whatever) comes in
- Publish-and-subscribe mechanism
  - Well known to Smalltalkers
  - Adapters in VisualWorks and VisualAge
  - Based on event model in the domain
    - MVC dependents

Links between hub and spokes

- Events out, messages in
- No direct dependencies between business component and “outside world”

Current work

- Of course, this architecture is not dependent on Smalltalk
- Currently implemented in Dutch Public Order and Security (mainly Police) with Java used for the hub
  - Java creates many problems
  - J2EE by long not ready for domain implementations
    - Too much focus on database connectivity
    - Too little support for active objects
    - Internal concurrency not allowed
- Management could not be convinced to use Smalltalk ☹

Thank you