Developing and managing dynamic business collaborations

A rule based approach for modeling and verifying business collaboration systems

Bart Orriëns
Department of Information Management
Tilburg University
... And everything after

- Introducing a new order of things
- Standing on the shoulders of giants
- Realizing that context = KEY
- Creating models that are useful
- Obeying the rules and only the rules
- Arranging everything in the right place
- Performing magic
- Reasoning towards conclusions
- Predicting the future
The most likely way for the world to be destroyed, most experts agree, is by accident. That’s where we come in, we’re computer professionals. We cause accidents;
So what do we need to know?

- What is the setting in which business collaborations take place?
- How do we describe this setting?
- What is the role of change in all this?
- Are we doing things in the right way?
Albert Einstein

The secret of creativity is knowing how to hide your sources;
It has all been done before

**Context**
- **Levels**
  - Business/technical, strategic/operational, CIM/PIM/PSM
- **Aspects**
  - private/public, global/local, interfaces/protocols
- **Other**
  - Security, quality payment, logging transaction, legal issues

**Aspects**
- **Change**
  - Meta-model/open-point, model generating
  - Backward/forward, fault handler
  - Abort/continue, migrate
- **Facets**
  - functional/data/control/organization/structure/behavior/information

**Facets**
- **Verification**
  - Constraint satisfaction
  - XSRL, e-contracts
  - Model checking
  - FSMS, Petri Nets, Actor Model, process algebra
- **Theorem proving**
  - Situation logic, linear logic programming

**Verifying**
- **Other Meta-model/open-point, model generating**

**Modeling**
- **BPM**
  - BPML, BPSS, Workflow, EPC
- **WS**
  - BPEL, WSDL, WS-TX, WSMF
- **E-contracts**
  - WS-Agreement, WS-Policy, CPA/CPP, WSLA, WSOL
- **Other**
  - WS-Security, XAML, WS-TX, And etceteras
- **Semantic**
  - DFD, CFD, OO, DAML-S, OWL
- **Formal**
  - Petri nets, Process algebras, State machines

**Formal Adaptability**
- WS-Security, XAML, WS-TX, And etceteras
But then maybe not quite

- No overall picture of context
- No coherent modeling approach
- No generic change mechanism
- No generic verification solution
We are searching for some kind of harmony between two intangibles: a form which we have not yet designed and a context which we cannot properly describe;
1) What are the activities we undertake, what resources do they use?

2) In what way does our process support what we offer?

3) How do our agreements relate to what we can offer?

4) How does our business relate to our IT?

5) How are payment, quality, security and so on arranged?
Paulina Porizkova

When I model I’m pretty blank. You can’t think too much or it doesn’t work;
Bernard Traven

Rules. We don’t need no stinking rules (adapted version);
1) What happens if we change the order of activities we undertake?

2) What happens to our protocol if we change our process?

3) Can we meet a proposed change in our agreement with AGFIL?

4) How can we verify that our business and IT are aligned properly?

5) How can we assess the effects of changes in our quality policy?
The magic bullet is made of rules

- Capturing requirements in an explicit and thus manageable form
- Ensuring conformance of models to given organizational requirements
- Verifying and validating business collaboration system models
Plans and Goals

Enterprize Level

Business Process Architecture

Business Process Level

Implementation

Procedural rules implemented in managerial and employee actions

Procedural rules implemented in applications and databases

Logical Level

Activity Management

Activity Measurement

Job Design

Application Architecture

Data Architecture

Network Architecture

Physical Level

Management Rules

Measurement Rules

Rules followed by employees

Human-IT Interface

Rules in Programs, etc.

Databases Rules

Rules on the Flow of Information

Planning, Budgeting, Hiring, Resource Allocation

Activity Mon. System

Job Aids & Training

Application Design and Code

Database Design and Data

Technology and Hardware Architecture

Human Resource Architecture

IT Architecture
A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools;
But how do we develop rules?

- **Classic rule issues:**
  - Ambivalence
  - Circularity (with negation)
  - Deficiency
  - Redundancy

- **Additional issues:**
  - Versioning
  - Life cycle management
  - Reuse
And once we have rules, how to apply them?

- Generate models:
  - Only use active rules
  - Flow based rule processing
  - Conflict resolution handling

- Check resulting models:
  - No parts missing
  - No incorrect information defined
  - No conflicting details specified
And what is our prize?

- Flexible at design and runtime:
  - Defining different rules leads to different models
  - Changing existing rules

- Adaptive at design and runtime:
  - Specifying rules to deal with expected events
  - Introducing rules to handle unexpected events
Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning;
Design Schema Manager (DSM)

Design Schema Analyzer (DSA)

Design Generator (DG)

File Based Repository (FBR)

Specifying

Analyze

Simulate

User Interface (UI)

Icarus

Relies on

Simulate

Analyze

Specify

OO jDREW Rule Engine

Design Schema

Model Schema

Model Schema

Design Schema

Model Schema

Model Schema
Edit policy alternative

- Create policy alternative
- Include policy alternative by reference
- Extend policy alternative

Name: NormalProcedure

Description: This alternative describes the normal procedure for how to handle a damaged car

Activation: 1 1 2007
Expiration: 1 1 2012

Guards:

Version: 1.0
A conclusion is the place where you get tired of thinking;
So what did we learn?

- Information systems for business collaborations are complex.
- They require extensive modeling covering multiple dimensions.
- The resulting models must be verifiable and manageable in order to cope with change.
Some more things we learned

- Rule based development and management seems viable
- Allows changes to be made and their effects to be managed
- Enables verification of business collaboration systems
The future, according to some scientists, will be exactly like the past, only far more expensive;
Are we there yet?

- Incorporate additional advanced requirements
- Introduce more perspectives
- More sophisticated rules may be needed
- Embrace semantic oriented solutions
Great is the art of beginning, but greater is the art of ending;