OOP and S-BPM – an analogy observation

PowerSpeech

Karlsruhe, October 14th 2010

Hagen Buchwald, KIT, Institute AIFB
Potential Building Blocks of S-BPM (S-BPM ONE 2009)

Technology

- **T1**: S-BPM Notation
- **T2**: S-BPM Architecture
- **T3**: S-BPM Reference Implementation

Community

- **C1**: S-BPM Publication Series
- **C2**: S-BPM ONE Conference
- **C3**: S-BPM Community Process

Methodology

- **M1**: S-BPM Patterns
- **M2**: S-BPM Process Life Cycle
- **M3**: S-BPM Maturity Levels

Personal Process Space
Why is a simple business process complex?
The Dice Game.
Why is a simple business process complex?

- Cycle Time
- Backlogs
- Throughput
- Costs
Why is a simple business process complex?

Throughput

Backlogs

Costs

Cycle Time
Why is a simple business process complex?

- Throughput
- Backlogs
- Costs
- Cycle Time
Why is a simple business process complex? The KreditSim role play.

State 0
- SB Rating Sicherheiten

State 1
- SB Rating Bonität

State 2
- SB Zinskalkulation

State 3
- SB Kreditentscheidung

State 4
- AL Kreditentscheidung

State 5
- SB Service

State 6
- AL Refinanzierung

State 7
- SB Ausg.-bearbeitung

S0 S7

The KreditSim role play.
Why is a simple business process complex? First answer: Process dynamics.

Variance

- Filiale
- SB Rating Siche
- SB Rating Bonitat
- SB Zinskalkulation
- SB Kreditentscheidung
- AL Kreditentscheidung
- SB Service
- SB Ausgangsbearbeitung

Maximum
oberes Quartil
unteres Quartil
Minimum
Why is a simple business process complex? First answer: Process dynamics.

Variance

![Graph showing variance for different processes with Filiale having the highest variance and AL Kreditentscheidung having the lowest variance.](image-url)
Why is a simple business process complex? First answer: Process dynamics.
Why is a simple business process complex?
First answer: Process dynamics.

Histograms in comparison
For understanding the complexity of a business process, you first have to explore its natural context (nCEA).

State 0  State 1  State 2  State 3  State 4  State 5  State 6  State 7

S0  SB Rating Sicherheiten  SB Rating Bonität  SB Zinskalkulation  SB Kreditentscheidung  AL Kreditentscheidung  SB Service  AL Refinanzierung  SB Ausg.-bearbeitung  S7

\[4 \times 2 \times 1 \times 2 \times 2 \times 4 \times 2 \times 2 = 512\]
First Observation

1. The State is a Structure – a business object.
2. Reality is complex – there is a combinatorial number of paths from start to end.
3. We need a natural context exploration approach (nCEA, Matthes Elstermann, 2010) to understand (model – simulate – analyse) the complexity of a process.

How does that help us to find an adequate definition for S-BPM?
Personal Process Space
The Key Concept to formalize S-BPM

WHAT

pre-conditions

behaviours

invariants

post-conditions

HOW

refinement

ground model
Excours: Abstract State Machines
Yuri Gurevich, mid 1980

- An **Abstract State Machine (ASM)** is a state machine operating on states which are arbitrary data structures.
- A state transition can only be executed if the **pre-conditions** for that transition are fulfilled.
- The state transitions involve only a bounded part of the state, described by **post-conditions**.
- and everything is **invariant** under isomorphisms of structures.
- Pre- and post-conditions describe the **WHAT**-part of a transition – the specification.
- The **How**-part – the implementation – is described by the **behaviour** of the data structure, which acts like an algebra, always guaranteeing the consistency of the invariant.
- An **Abstract Data Type (ADT)** is an algebra describing the **WHAT**-part of an arbitrary data structure. It is the core concept of **OOP**.
- Thesis #1: **ASMs** are an excellent candidate as core concept of S-BPM.
- Thesis #2: The **states** of these ASMs must be described as **ADTs**.
OOP Maturity Levels

**Level 1**
Object-based, modular structure
Systems a modularized based on their data structures.

**Level 2**
Data abstraction
Objects must be described as implementations of abstract data types (ADTs).

**Level 3**
Garbage collection
Unreferenced objects should be deallocated automatically by the underlying runtime system.

**Level 4**
Classes
Classes are implementations of abstract data types (ADTs).

**Level 5**
Inheritance
A class can be defined as an reduction or extension of another class.

**Level 6**
Polymorphism and dynamic binding
Elements of a system may reference objects of more than one class, and routines may have different implementations in different classes.

**Level 7**
Multiple inheritance
You may declare classes which inherit from more than one parent class.

Source: Bertrand Meyer, Object-Oriented Software Construction, 1986
## S-BPM Maturity Levels

| Level 7 | Multiple inheritance  
|         | You may declare classes which inherit from more than one parent class. |
| Level 6 | Polymorphism and dynamic binding  
|         | Elements of a system may reference subjects of more than one class, and behaviour may have different implementations in different classes. |
| Level 5 | Inheritance (Refinement)  
|         | A class can be defined as an reduction or extension of another class. |
| Level 4 | Classes  
|         | Classes are implementations of abstract state machines (ASMs). |
| Level 3 | Linear scalability  
|         | The more cores the underlying hardware offers, the more subjects can act in parallel in order to respond to higher market demand. |
| Level 2 | Behaviour abstraction  
|         | Subjects must be described as implementations of abstract state machines (ASMs). |
| Level 1 | Subject-based, modular structure  
|         | Systems a modularized based on their subject's behaviour (habits). |

Source: Bertrand Meyer, Object-Oriented Software Construction, 1986
S-BPM-Systems are Software Systems.
But what is a software system?

Source: Bertrand Meyer, Object-Oriented Software Construction, 1997
S-BPM-Systems are Software Systems. But what is a software system?

A software system consists of processors, which execute methods on data structures.

The difference between OOP and S-BPM is our primary question to the system:

OOP: On what does the system work?
S-BPM: Who acts in the system?
S-BPM-Systems are Software Systems.
But what is a software system?

A S-BPM system consists of subjects, which execute predicates on objects.

The difference between OOP and S-BPM is our primary question to the system:
OOP: On what does the system work?
S-BPM: Who acts in the system?

Source: Albert Fleischmann, S-BPM ONE 2009
What is OOP?
A formal definition.

A software system consists of processors, which execute methods on data structures.

OOP is the development of software systems as a structured collection of implementations of Abstract Data Types (ADTs).

Source: Bertrand Meyer, Object-Oriented Software Construction, 1997
What is S-SPM?
A proposal of a formal definition.

A software system consists of processors, which execute methods on data structures.

S-BPM is the development of software systems as a structured collection of implementations of multi-agent Abstract State Machines (ASMs) with ADTs as states.
Each operative class is completely described by its Interface and the corresponding ContractClass.

**WHAT**

Object-oriented Programming is the development of software systems as structured collections of Implementations of abstract data types (ADT).

**HOW**

Specification Level
- Abstract Data Type

Implementation Level

```java
<<Interface>>
BesprechungSpec
+ anlegen (java.lang.String, java.lang.String, java.util.Date, java.util.Date, java.util.Date) : void
```

```java
<<Interface>>
Besprechung
+ anlegen (java.lang.String, java.lang.String, java.util.Date, java.util.Date, java.util.Date) : void
+ anlegen (java.lang.String, java.lang.String, java.util.Date, java.util.Date, java.util.Date) : void
```

```java
BesprechungTest
- m_target : Besprechung
  - (frozen) name : java.lang.String
  - (frozen) thema : java.lang.String
  - (frozen) start : java.util.Date
  - (frozen) ende : java.util.Date
  + setupTest () : void
  + tearDownTest () : void
  + anlegenTestNichtStartEnde () : void
  + anlegenTestErrorConditionStartEnde () : void
  + anlegenTestErrorConditionStartDatumBeforeEnde () : void
```
Interfaces specify the syntax part, Contract-Classes the semantic part of the ADT.
Results of our Design by Contract experiments SS2010
Group A (Interface)

Story Card  ASpec.java
Results of our Design by Contract experiments SS2010
Group B (Interface and Test-Class)
Results of our Design by Contract experiments SS2010
Group C (Interface, Test-Class and Contract-Class)

- Story Card
- ASpec.java
- ATest.java
- ASpecContract.java
What is S-SPM?
A proposal of a formal definition.

A software system consists of **processors**, which execute **methods** on **data structures**.
Conclusion

**Technology**

- **T1** S-BPM Notation
- **T2** S-BPM Architecture
- **T3** S-BPM Reference Implementation

**Community**

- **C1** S-BPM Publication Series
- **C2** S-BPM ONE Conference
- **C3** S-BPM Community Process

**Methodology**

- **M1** S-BPM Patterns
- **M2** S-BPM Process Life Cycle
- **M3** S-BPM Maturity Levels

---

**Comprehensive ASM Specification of PASS**

Egon Börger

---

**Personal Process Space**