Learning Objectives

- Become aware of quality issues
- Understand the role of anti-patterns and patterns in process modeling

- Exercise: Simulation of typical control-flow errors
  - Lack of Synchronization
  - Deadlock

- Demo: Pattern-based Process Model Accelerators
  - patterns vs. transformations vs. refactorings

- Exercise: Patterns in the T Order management process
Quality of Business Process Models is becoming an increasing Concern

- Business process modeling moves from informal documentation towards the specification of SOA solutions
  - more people with different background get involved into modeling projects
  - process models become a direct input into the software-development process
    - paradigm of Business-Driven Development
- Impact of process models on the IT systems and the operational efficiency of an enterprise is increasing
  - associated economic risk is growing

- Quality assurance required
- Find errors in model, not when implemented solution is tested

Where do we stand?

- Process modeling tool vendors teach users how to use a tool and show simple process samples
  - users do not learn about potential errors and how to avoid them
  - users do not learn how to create “good” process models
- Review of models created in various tools shows that almost every model that tries to describe a realistic process is wrong
  - users get simple sequences of tasks correct
  - as soon as decisions, cycles or parallelism occur in a process model, the chance for an error is about 50 %
- Academia: very few courses on modeling – usually do not teach how to create a good model
Process Model Quality Assessment: State of the Art

- Few results that really help users to produce models of better quality

  - Correctness, relevance, economic efficiency, clarity, comparability, systematic design
  - No direct measures given

- Adoption of ISO/IEC 9126 Software Product Quality Model for process models [Selcuk Guceglioglu & Demirors, 2005]
  - Functionality, reliability, usability, maintainability
  - Based on subjective ratings by human users

- Well-formed Workflow Nets [Aalst & Hee, 2002]
  - strongly limited structures to avoid certain types of errors

Paradox of Behavioral Modeling

- Process models describe behaviors

- behaviors can also be described in programming languages
  - these languages are very structured
  - structure helps users to avoid errors and enables compilers to spot many errors (not all, though)

- Business process modeling languages are unstructured
  - they offer many powerful (& unstructured) modeling constructs to express any kind of behavior
  - current education introduces them all without alerting users of what can go wrong

- Business users are not well-enough trained in behavioral modeling
Anti-patterns

- “classes of commonly reinvented bad solutions to problems” (Wikipedia)
- Problem: create a correct model of a process
  - Anti-patterns capture typical design errors in a process model, which make the process model incorrect
  - allow us to measure an important aspect of the quality of process models in an objective way – correctness
- Soundness as a major prerequisite for correctness
  - anti-patterns that identify deadlocks and lack of synchronization errors

Methodological Approach

Errors collected from hundreds of real-world models

Abstract example illustrating “typical” error

Anti-pattern followed by the correct solution
What do we mean by a “good” Model?

- models are executable, e.g. by simulation
  - they describe the possible behaviors of a process
  - these behaviors result in execution traces that we can observe in the simulation

- a good model should show the intended process behaviors
  - it should also be well-structured and make it easy for the user to understand the process behavior

What can go wrong when modeling a process?

- the model can lead to more behaviors than intended
  - we obtain more instantiations of tasks than we thought
  - usually caused by a lack of synchronization
  - a livelock can lead to infinite iterations of a process making no progress

- the model can lead to less behaviors than intended
  - certain tasks do not execute (in some or all traces)
  - usually caused by a deadlock
Deadlock and Lack of Synchronization

- **Deadlock**
  - Process is blocked forever, some activities never execute.

- **Lack of Synchronization (LoS)**
  - Results in multiple executions of activities (and risks of uncontrolled data access).

*Sound process* = No deadlocks + No lack of synchronization

Common Control-Flow Anti-patterns when modeling branching Behavior

- Lack of Synchronization through AND-XOR

- Potential lack of Synchronization through IOR-XOR

- Deadlock through XOR-AND
Common Control-Flow Anti-patterns when modeling cyclic Behavior

- Cyclic Deadlocks through AND-AND and AND-XOR
  - Deadlock

- Cyclic Lack of Synchronization through XOR-AND
  - Deadlock

- Potential Lack of Synchronization through XOR-IOR
  - Potential LoS

Exercise

- Behavior of lack of synchronization and deadlock errors when simulating processes in WebSphere Business Modeler
BPMN-specific Anti-patterns

Source: Polančić & Rozman
University of Maribor, Slovenia
http://bpmn.itposter.net

BPMN-specific Anti-patterns II

Improperly used Gateways

Gateways are connected only with sequence flows. Avoid also potential deadlocks when using gateways.

Improper modelling of Pools

When modelling Pools, sequence flows and start/end events are often missing, because it is wrongly presumed that message flows substitute sequence flows. Additionally, sequence flows are incorrectly used to connect pools.

Model the process in each Pool independently and afterwards define message flows between Pools.
BPMN-specific Anti-patterns III

Improper use of Lanes

Lanes are often wrongly used in similar ways as Pools. They wrongly contain more business processes or contain message flows between different lanes.

BPMN-specific Anti-patterns IV

Incorrect sequence flow mechanism

When using expanded sub-processes, sequence flows should be connected to the boundaries of sub-processes. Processes and sub-processes should start and end properly.

Sequence flow cannot cross the boundary of sub-process

The sub-process should have a start event

The process should have an end event
BPMN-specific Anti-patterns V

Improper tasks and events

Analyzers often wrongly model events and tasks. For example: events are wrongly modeled as tasks; task states are modeled as new tasks.

- Task automatically starts at input sequence flow
- Task A is automatically finished at output sequence flow
- The getting of document is represented by incoming message flow

BPMN-specific Anti-patterns VI

Improper use of message events and message flows

Starting and intermediate events cannot be sources of message flows.

- Both examples are wrong - intermediate message events can not produce message flows
- Wrong positioning of message event
Patterns

- A pattern encapsulates a simple and elegant solution to a specific, but frequently re-occurring problem
- Patterns are created by observing (or “mining”) a variety of solutions that different people have created over time when working on the same problem
- The pattern encapsulates the “essence” that is common to all these different solutions and that ensures that the solution solves the problem

- Examples of famous collections of patterns
  - object-oriented design patterns (Gamma et al)
  - software architecture patterns (Buschmann et al)
  - workflow patterns (van der Aalst et al)

Workflow Patterns for Control-Flow (Aalst et al)

- Available in most modeling tools
- Too fine grained - introduce many modeling errors
  - see anti-patterns
Correct Model Fragments (Patterns)

- Correct Branching through XOR-XOR and AND-AND and IOR-IOR

- Correct Cycles with XOR-XOR

Overlapping Pattern
Beyond Patterns: Transformations & Refactoring

- Patterns make it easier to correctly model a process “from scratch”
- However, changes occur very frequently
  - hard to apply in a drag-and-drop tool
  - model becomes cluttered and hard to understand

- Automate change application with model transformations

- Leverage refactoring from software engineering in model-driven tools

(Model) Transformation

- Automatic generation of a target model from a source model according to a transformation definition (Kleppe et al)
  - transformation definition: set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language
  - transformation rule: description of how one or more constructs in the source language can be transformed into one or more constructs in the target language

- Endogenous transformation: between models expressed in the same language (vs. Exogenous)

- Horizontal transformation: source and target models reside at the same abstraction level (vs. Vertical)
Refactoring

- **Refactoring** (noun): A small change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior (Fowler)
- **Refactor** (verb): to restructure software by applying a series of refactorings without changing its observable behavior (Fowler)

Improve the quality of software:

- “Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.
  - It is a disciplined way to clean up code that minimizes the chance of introducing bugs. In essence when you refactor, you improve the design of the code after it has been written. With refactoring you can take a bad design, chaos even, and rework it into well-designed code.
  - Each step is simple, even simplistic. Yet the cumulative effect of these small changes can radically improve the design.”

Demo

- IBM Pattern-based Process Model Accelerators for WebSphere Business Modeler

<table>
<thead>
<tr>
<th>Control-Flow Patterns</th>
<th>Transformations</th>
<th>Refactoring Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Insert Task</td>
<td>- Associate Data…</td>
<td>- Merge Process Ends</td>
</tr>
<tr>
<td>- Insert Process…</td>
<td>- Autolink Elements</td>
<td>- Join Process Ends</td>
</tr>
<tr>
<td>- Sequence…</td>
<td>- Toggle Gateways</td>
<td>- Close Branches</td>
</tr>
<tr>
<td>- Alternative Compound…</td>
<td>- Duplicate Element</td>
<td>- Automatically Order Branches</td>
</tr>
<tr>
<td>- Parallel Compound…</td>
<td>- Merge Elements</td>
<td>- Activity to Gateway Form</td>
</tr>
<tr>
<td>- Loop…</td>
<td>- Replace Global Process…</td>
<td>- Extract Subprocess</td>
</tr>
<tr>
<td>- Alternative Branch</td>
<td>- Remove Region</td>
<td>- Inline Subprocess</td>
</tr>
<tr>
<td>- Parallel Branch</td>
<td>- Reorder Sequence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Convert to Alternative Compound</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Convert to Sequence</td>
<td></td>
</tr>
</tbody>
</table>
Background Reading


Summary Part 3

- Quality assurance for business process models becomes increasingly important
  - no longer used for documentation only, but to specify requirements for business process transformation and implementation
    - economic impact of modeling errors is dramatically increasing
    - more people involved in modeling process

- Our focus for this course:
  - correctness of control-flow (soundness) as essential quality measure

- Control-flow patterns and anti-patterns