Workflow Modeling & Verification

- Topics
  - Modeling Perspectives
  - Workflow Languages
  - Workflow Verification

- Readings
  - Lecture Notes

Workflow Modeling

- Many languages invented with individual product developments
- Petri Nets - most research papers
- Many complex issues on expressability and verifiability
- Little impact of research on products

Workflow Modelling Perspectives

<table>
<thead>
<tr>
<th>Primary</th>
<th>Secondary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>Security</td>
</tr>
<tr>
<td>Informational</td>
<td>Causality</td>
</tr>
<tr>
<td>Behavioural</td>
<td>History</td>
</tr>
<tr>
<td>Operational</td>
<td>Integrity</td>
</tr>
<tr>
<td>Organizational</td>
<td>Quality</td>
</tr>
</tbody>
</table>

Source: Jablonski and Bussler (1996)

Primary Modelling Perspectives

- Functional
  - What is to be performed
  - Activities and Sub-processes

- Informational
  - What information is required
  - Input and Output data
### Primary Modelling Perspectives

**Behavioural**
When it is performed
- Control Flow
  - Dependency between workflow activities
  - Typical structures that represent this dependency:
    - Sequential execution
    - Concurrent execution
    - Alternative execution

**Temporal Constraints**
- Durations (Single, Interval)
- Deadlines (Absolute and Relative)
- Interdependent time limits

**Operational**
How it is performed
- Workflow Relevant Applications
  - Degree of modification (workflow aware applications)
  - System support (manual or automated)
  - Granularity (functional complexity)
  - Scope (manipulating workflow control data or only application data)
  - Coupling (integration between application and WFMS)
  - Invocation mode (synchronously or asynchronously)
  - Interaction mode (interactive or batch)

### Secondary Modelling Perspectives

**Security**
Who is allowed access
- Potential conflict with selection policies

**Causality**
Does the model abide by the business policies, rules and strategies

**History**
What happened during execution
- Audit trail/workflow logs
  - System context of Audit trail
    - Queries on history (who performed this activity)
    - Failure and recovery (determine last consistent state)
  - Application context of Audit trail
    - Analysis (temporal constraints, participant loads, exceptions)
    - Evolution and Improvement (semantic failures)

### Workflow Languages

**Formal Languages**
- Petri Nets
- Process Algebra
- Computational Tree Logic

**Commercial Languages**
- Flowmark (MQSeries), Forte, Action, Staffware, …
- See Workflow Vendors at [www.waria.org](http://www.waria.org)

**Research Proposals**
- Adept, Wide, Mobile, METEOR, TRAMs, WAMO, …
- See Workflow Prototypes at [http://www.dbtg.unizh.ch/Links/workflow.html](http://www.dbtg.unizh.ch/Links/workflow.html)

**Standards**
- Workflow Management Coalition
- Business Process Modelling Initiative
- Web Services Definition Language
- XLANG Business Process Language
- …
Petri-Nets based Workflow Models

- WF Nets [Van der Aalst W.M.P.]
- Information Control Nets [Ellis C.A., Nutt G. J.]
- Modular Process Nets [Wikarski D.]
- Reconfigurable Nets [Badouel E., Oliver J.]
- Higher Order Object Nets [Wikarski D., Han Y., Lowe M.]

Source: Gerrit K. Janssens, Jan Verelst, Bart Weyn. (2000)

A Workflow Language

- Generic language with a small number of constructs
- Sufficient expressiveness for a wide variety of process requirements
- Simplicity provides rigorous analysis and verifiability
- Developed jointly by DSTC and UQ researchers (1995 – 2001)
- Supported by a process modelling and verification tool

Fundamental Modeling Aspects

- Structure
  - Control Flow
- Data
  - Input and Output
- Time
  - Deadlines and Durations
- Resources
  - Applications, Roles, Performers

Core Structures

- Sequence
- And Split
- And Join
- Or Split
- Or Join
- Nesting
- Iteration
- Termination

Example

Alternative Modelling Approach

- More intuitive
- Less graph nodes
- Implicit semantics

Mapping from Explicit to Implicit Representation
Extended Modelling Structures

- Multiple choice
- Multiple merge
- N-out-of-M join
- Implicit Termination

Multiple Choice

The choice of branches after an or-split is not exclusive

Multiple Merge

The merge can have more than one incoming branch
Subsequent activities will be activated as many times

N-out-of-M Join

Waits for a given number of branches and then ignores (cancels) the remaining

Implicit Termination

Process is complete when there are no active activities left

Explicit, unique terminating node

{B, D, E} represent terminating nodes (activities)

Exercise 1

Think of examples where the core modelling structures would not be sufficient
Language Syntax

Core Modelling Structures for Explicit representation

- Let $G = \langle N, F \rangle$ be a graph where $N$: Finite Set of Nodes, $F$: Flow Relation $F \subseteq N \times N$
- $\forall n \in N$, NodeType: $n \mapsto \{\text{Coordinator, Task}\}$
- $N = C \cup T$, $C \cap T = \emptyset$ where $C$: Set of Coordinator Nodes, $T$: Set of Task Nodes
- $\forall n \in C$, CoordType: $n \rightarrow \{\text{Fork, Synchronize, Choice, Merge, Begin, End, Do, While}\}$
- Let $P$ be a directed path in $G$, such that $P = \{n_1, n_2, …, n_k\}$, $(n_i, n_{i+1}) \in F$ for $i = 1, 2, …, k - 1$
- Let $\text{Nat}$: Set of Natural Numbers, $\text{NId}$: Set of Node Identifiers
- $\forall n \in N$, $\text{I}: N \rightarrow \text{Nat}$, $\text{I}(n)$: Number of incoming flows for node $n$
- $\forall n \in N$, $\text{O}: N \rightarrow \text{Nat}$, $\text{O}(n)$: Number of outgoing flows for node $n$

Structural Specification

A Workflow is a Directed Acyclic Graph (DAG) $W = \langle N, F \rangle$ such that

- $\exists n \in C, \text{I}(n) = 0 \wedge (\neg \exists m \in N, \text{I}(m) = 0 \wedge m \neq n)$, we call this Begin Node $n_0$ and CoordType $(n_0) = \text{Begin}$
- $\exists n \in C, \text{O}(n) = 0 \wedge (\neg \exists m \in N, \text{O}(m) = 0 \wedge m \neq n)$, we call this End Node $n_f$ and CoordType $(n_f) = \text{End}$
- $\forall n \in T$, $\text{I}(n) + \text{O}(n) > 1$ where $n \neq n_0$ and $n \neq n_f$
- $\forall n \in T$, TaskType: $T \rightarrow \{\text{Activity, SubProcess}\}$, Activity represents a single task and SubProcess represent nesting.

Basic Meta-model for Process

Basic Meta-model for Activity

FlowMake

- Modelling, Analysis and Verification of Workflow models
- Objectives
  - Simple modelling language
  - Correctness criteria
  - Verification algorithms
  - Modelling and verification tool

Download available from course website
**FlowMake Components**

- Workflow Editor
- Verification Engine
- Persistence Interface
- Relational Database
- C++ Object Serialization
- IBM FlowMark
- Mincom MHC/Vone
- Enterprise Interface
- Product Interface
- Canvassing Tools Library
- Participant Assignment
- Application Handler

**FlowMake: Demonstration**

- Developed in Microsoft Visual C++
- Fast and compact
- User-friendly WF Editor
- Verification and Analysis
- Object-oriented design
- Easily extensible
- Implements DSTC research
- Product interfaces

**Exercise 2**

Make a new model in FlowMake for (any or all)

- Library loans
- Insurance claims
- Car rental
- University admission

**Workflow Verification**

- Semantic Verification
  - Verify that the model is in conformance with the business process goals
- Syntactic Verification
  - Verify that the model is in conformance with the grammar of the language
- Structural Verification
  - Verify that the model will not lead to erroneous execution

**Syntactic Errors**

- An activity node cannot have more than one incoming/outgoing flows (explicit representation)
- Reachability of nodes (Graph must not be disconnected)
- Multiple (initial) final activities

**Structural Errors**

- Mostly represent errors in control flow specification
- Incorrect specification of data, time and resources will also generate error in execution
Data, Time and Resource Conflicts

• Data Conflicts
  – Missing data
  – Lost data

• Temporal Conflicts
  – Temporal Consistency
    • Build Time
    • Run Time

• Resource Conflicts

Missing data:
Where is Y coming from

Data, Time and Resource Conflicts

• Data Conflicts
  – Missing data
  – Lost data

• Temporal Conflicts
  – Temporal Consistency

• Resource Conflicts

Lost data:
What will be the value of X

Data, Time and Resource Conflicts

• Data Conflicts
  – Temporal Consistency
  • Build Time
  • Run Time

• Resource Conflicts

Duration Constraint:
Minimum time is 7 hours

Interdependent Constraint:
Must be completed within 3 hours after completion of task A

Data, Time and Resource Conflicts

• Data Conflicts
  – Temporal Consistency
  • Build Time
  • Run Time

• Resource Conflicts

Duration Constraint:
Minimum time is 3 days
<Started on 1/1/02>

Deadline Constraint:
Must be completed by 2/1/02

Data, Time and Resource Conflicts

• Data Conflicts
  – Temporal Consistency
  • Build Time
  • Run Time

• Resource Conflicts

Duration Constraint: Minimum time is 2 days
<Started on 1/1/02>

Deadline Constraint: Must be completed by 2/1/02

Data, Time and Resource Conflicts

• Data Conflicts
  – Incomplete specification
    • Role is assigned to activities, but no participants are bound to that role
  – Access and Role Conflicts
    • Participant does not have access to activity A, but is assigned a role that can perform activity A
  – Other … ?

Exercise 3

Any other data, time or resource conflicts?
Control Flow Conflicts

- Deadlock: Synchronizing alternative paths
- Lack of synchronization: Merging concurrent paths

Lack of Synchronization

Correct Workflow Models

Control Flow Verification

Based on the concept of an instance subgraphs
- An instance sub-graph represents a subset of nodes (workflow tasks) that may be executed for a particular instance of a workflow
- It can be generated by visiting the nodes of a workflow graph on the basis of the semantics of underlying control flow constructs
- In the language under consideration, the "Choice" construct produces more than 1 instance sub-graph
- If there is one choice construct, then number of instance sub-graphs would be = number of outgoing flows of the choice coordinator
- Number of instance graphs increase exponentially with the number of choice-merge constructs
- A brute force method of generating all possible instance sub-graphs is not computationally effective

Instance Sub-graphs
Correctness Criteria

- Criteria 1: Deadlock free workflow graphs
  - A workflow graph is free of deadlock structural conflicts if it does not generate an instance sub-graph that contains only a proper subset of the incoming nodes of a synchronizer node.

- Criteria 2: Lack of synchronization free workflow graphs
  - A workflow graph is free of lack of synchronization structural conflicts if it does not generate an instance sub-graph that contains more than one incoming node of a merge node.

CF Verification based on Reduction

- Remove all such structures within a workflow graph that are definitely correct
- A conflict-preserving reduction process is iteratively applied
- A structurally correct graph would reduce to an empty graph
- A workflow graph containing structural conflicts is not completely reduced

Reduction Rules

- Terminal Reduction
- Sequential Reduction
- Adjacent Reduction
- Closed Reduction
- Overlapping Reduction

Applying Reduction Rules

Reduction Algorithm
Reduction Algorithm

Reducing a Structurally Incorrect Workflow Graph

Next

Workflow Management