Business -- IT Gap

“to-be” process models
- modeled by domain experts
- communicate, prescribe, illustrate
- shared understanding between participants
- intuitive, abstract, ambiguous, need some interpretation

Executable process models
- modeled by IT experts
- input to a process engine
- formal executable semantics
- explicit, concrete, precise and full of implementation details

Automation Goals

Repeatable Processes: processes are executed in the same way, tasks may be manual

Straight through processing: full process and task automation, high throughput

Overview
Two-level Programming

1. Programming in the large
   - Process logic defines the integration of reusable components (control and data flow)
   - Assembly by domain experts (non-programmers)

2. Programming in the small
   - Component logic implements discrete fine-grained functions and tasks (interfaces)
   - Components developed by IT experts (programmers)
APIs are used to access software functionality exposed to be integrated from processes.

If APIs are not available, it is always possible to access a system from its user interface, or go directly to its data.

Each participant follows his own process when interacting directly with the other software systems involved in the choreography.

The process defines the behavior of a centralized coordinator, which will drive the interaction between all integrated software systems.
**Composition**

The process defines the behavior of a component that is (recursively) built out of the composition of other components, which remain hidden from the client.

**Megaprogramming**

With a process, every instruction:
- Can dynamically select, bind and execute a different external software system
- Can timeout and be aborted if it doesn’t complete within a deadline
- Can be part of a distributed atomic transaction
- Is logged persistently so that its execution state can be recovered and monitored
- Can be suspended, resumed, canceled or retried by the user

**Process-aware Software**

- **Worklist Handler** - the user interface for notifying users about new tasks to be executed and for notifying the engine that users have completed their tasks
- **Embedded Software Activities** small-grained functions that are embedded efficiently into the process execution thread (e.g., automated decision rules, data conversion operators)
- **Remote, Standard Software** invoked through a standardized protocol (HTTP, WS-* SOAP)
- **Remote, Non-Standard Software** requires the development of custom adapters before it can be invoked from the process engine
**Software Connectors**
- Remote Procedure Call
- Asynchronous Messaging
- File Transfer
- Shared Database
- Stream

**Connectors and Processes**
- Remote Procedure Call
- Asynchronous Messaging
- File Transfer
- Stream
- Process

**Internal Engine Architecture**
- Process Navigator
- Process Designer
- Process Monitor
- Task Dispatcher
- Available Resources
- Completed Tasks
- Pending Tasks
- Assigned Tasks
**Correlation**

Problem: Route incoming messages to the corresponding existing process instance or start a new process to process an incoming message

Solution: Embed process identifier into message headers

Alternative Solution: Use message properties and content to identify the process instance

---

**Process Identification**

- Package
- Process Name
- Version
- Instance Counter
- Replica

Structured process instance identifiers allow to group related instances and find which process template they execute

---

**How long does it take?**

**Macroflow** long-running business processes (days, months, years) which may include human tasks

**Microflow** short-lived transactional processes (microseconds, seconds, minutes) that involve fully automated tasks

---

**Types of Process Engines**

- Process Representation:
  - Generic (Explicit) vs. Hard-Coded (Implicit)
- Process Modeling Language:
  - Standard (BPMN, WS-BPEL) vs. Custom
- Deployment:
  - Standalone vs. Embedded
**Commercial Engines**
- IBM WebSphere Process Server
- Oracle BPMS
- Microsoft BizTalk, Windows Workflow Foundation
- SAP NetWeaver BPM
- Software AG webMethods
- Appian BPMS
- BizAgi BPM Suite
- Bosch inubit Suite
- OpenText BPM
- Perceptive BPMONe
- Progress Savvion
- TIBCO ActiveMatrix BPM
- Whitestein Living Systems Process Suite

**Open Source Engines**
- Activiti
- Camunda
- Stardust
- Apache ODE
- Bonita
- Intalio BPM
- JBoss jBPM
- YAWL
- JOpera

**References**
- Carsten Hentrich, Uwe Zdun, *Patterns for Process-Oriented Integration in Service-Oriented Architectures*, EPLOP 2006