Data-aware Processes: Modeling, Mining, and Verification

Part 1: Modeling

Diego Calvanese

Research Centre for Knowledge and Data (KRDB)
Free University of Bozen-Bolzano, Italy

3rd International Winter School on Big Data (BigDat 2017)
13–17/2/2017 – Bari, Italy

"Sometimes I just feel like processing some data, but I have no data to process—other times I have the data, but I have nothing to process it with."

Outline

1. Data and processes: two sides of the same coin
2. Data-centric processes
3. The Guard-Stage-Milestone meta-model
4. Data-Centric Dynamic Systems

Outline

1. Data and processes: two sides of the same coin
2. Data-centric processes
3. The Guard-Stage-Milestone meta-model
4. Data-Centric Dynamic Systems

Processes and data

The information assets of an organization:

- **data**
- **processes**, determining how data changes and evolves over time

Survey by Forrester: Which of the two aspects should be given priority from the point of view of IT management? [Karel, Richardson, and Moore 2009]:

- Business process management professionals: view data as subsidiary to processes manipulating them, and neglect importance of data quality.
- Data management experts: consider data as the driver of the organizational processes and are concerned about data quality only.

There is still a dichotomy

Dichotomy in the relative perception of importance has a negative impact:

- Little collaboration between the teams
  - running the master data management initiatives, and
  - managing the business processes.

  Forrester: 83% . . . no interaction at all

- Little attention on the side of tool vendors to address the combined requirements:
  - Data management tools consider only the processes directly affecting the data in the tools, but not the actual business processes using the data,
  - Business process modeling suites do not allow for direct connection of data.

However, data and processes are tightly coupled together!

Two sides of the same coin

The need for overcoming this dichotomy is recognized by the (business) process modeling community as well [Reichert 2012]:

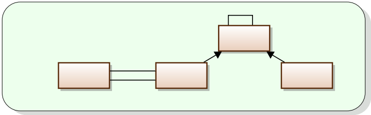
"Process and data are just two sides of the same coin"

Two key areas where explicit representation of data in business processes is important [Meyer, Smirnov, and Weske 2011]:

1. Modeling the core assets of an organization.
    - Data is crucial for the execution of business processes that create value.
    - Hence the business processes need to access the data, and this should be accounted for explicitly.
2. Business process controlling.
    - Key performance indicators and business goals are defined in terms of data.
    - To evaluate and control them, the data objects relevant for the activities contributing to the goals need to be identified.

Conventional data modeling

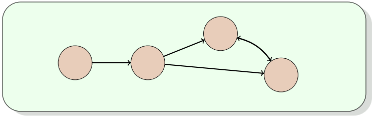
- Produce a structural model of the domain of interest
- Focus: entities, relations, and static constraints
- Formalisms: UML, ER, ORM, . . .
- Result: conceptual model, domain ontology, database schema



But how do data evolve?
E.g.: Where can we see the “state” of an order?

Conventional process modeling

- Produce a model of the dynamics of the domain of interest
- Focus: control flow of activities realizing the business objectives
- Formalisms: BPMN, UML AD, . . .
- Result: executable process model



But how are data manipulated?
E.g.: What impact does the cancellation of an order have?

Activity-centric process management

Main focus of consolidated modeling languages and process management systems: specification and execution of **predictable**, **structured**, and **repetitive** processes

Mainstream process modeling notations (e.g., BPMN) adopt **activity-centric** *procedural* and *imperative* approach

- coordinated execution of a set of business functions represented as atomic process steps/activities
- explicit specification of the activities to be performed and the execution relationships between them
  - process models structured in terms of flows of events and activities
  - focus is on control-flow perspective
- main driver for process progression: activity completion
  - activity completions enable subsequent tasks according to the control-flow
- successful process completion typically corresponds to the execution of defined tasks according to the prespecified ordering

Data perspective in activity-centric approaches

In activity-centric process models, the information perspective includes:

- a set of data objects and the data flow between activities
- the definition of which activities may read/write data elements as input/output parameters

Data manipulation is captured by means of:

- global variables defined within the scope of a process or subprocess, or
- conceptually passive data objects that are created, read, and/or updated by the events and activities in the process

The information and data flows are hidden in the model, and business data is mostly "unknown" to the process engine.

Data in activity-centric process-aware information systems

Activity-centric process-aware information systems (PAISs) distinguish between:

Application data:
: Managed out of the scope of the process by application services invoked during activity executions.

Process-relevant data:
: Represented as process variables that are read and updated by the activities, and used by the system to evaluate transitions and path choices (as routing conditions) within process instances.

Process control data:
: Define the current state of a process and its execution history.

Business objects and their attributes, encapsulating data and/or behavior, and managed by database applications and/or enterprise systems, **are outside the control of the PAIS**.

Example: Build-to-order process

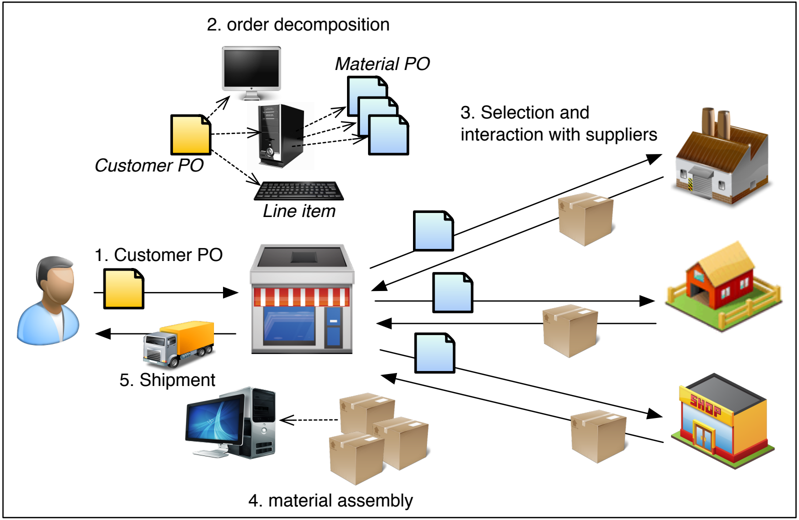
Build-to-order process

Products are manufactured/assembled on the basis of a confirmed purchase order (PO) issued by the customer.

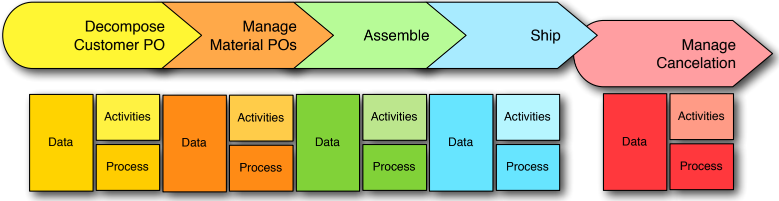
Overview:

1. Customer sends purchase order
   - A customer purchase order may contain one or multiple line items.
   - Each line item refers to a different product.
2. One work order is created for each line item.
3. Required raw materials/components are identified and ordered from a supplier.
4. Product is assembled and packed.
5. Products are shipped to the customer.

Example: Build-to-order process



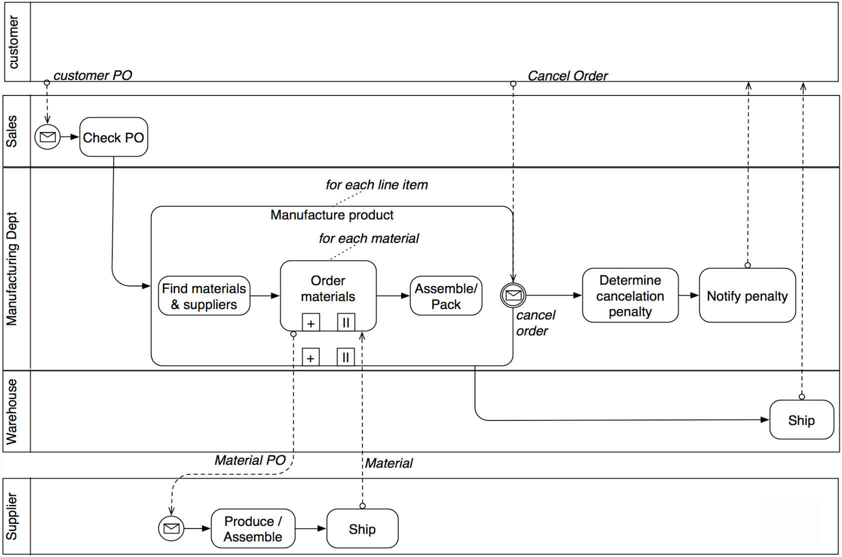
Traditional approach and decomposition



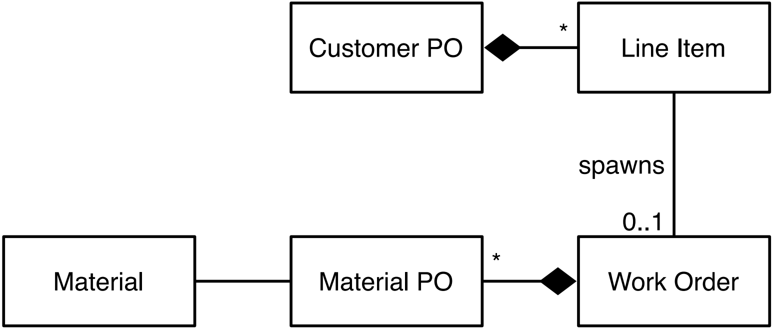
- Value chain construction
- Break-down of each phase into business functions
- Further decomposition:
  - data component (domain description)
  - activities (units of work) + process component (control-flow)
- **Impedance mismatch**: data and process divide!

High-level BPMN model

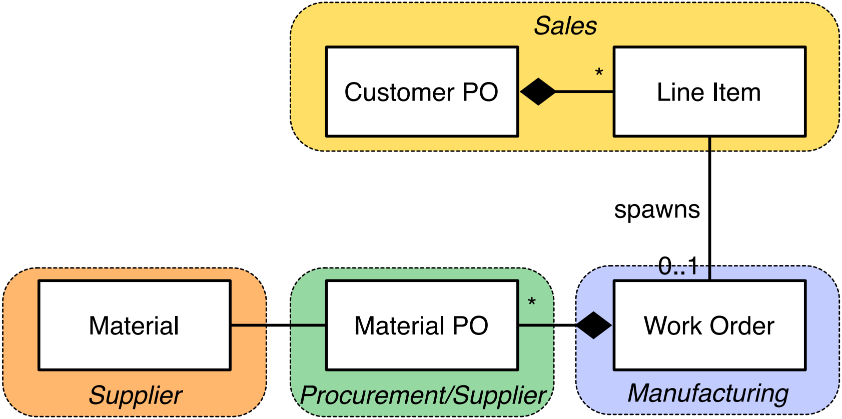
Control flow + Resource perspective



Data model


Customer PO
*
Line Item
spawns
0..1
Material
Material PO
*
Work Order

Distribution of responsibility



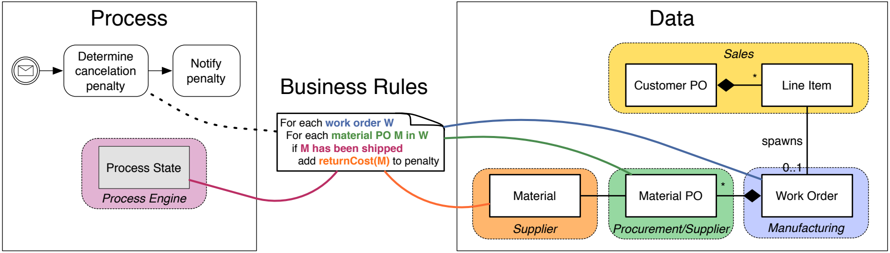
Problem: separation between data and processes

- At the modeling level: separation between data and processes.
- At the technological level: the connection is handled
  - ⤳ Lack of a conceptual view of data+processs that is holistic and coherent.
  - ⤳ It becomes difficult to manage and maintain the system.

Lack of a holistic view of data+processs makes it difficult to:

- Reconstruct and aggregate the relevant information:
  - part of the data is in the enterprise DBs
  - part of the data is hidden in the PAIS
- Decide how and where to model the domain relevant business rules:
  - At the enterprise DB level?
    - In which DB?
    - How to import the process data?
  - (Also) in the business model?
    - How to import data from the DBs?

Example: Cancellation business rule in “build-to-order”



```
For each work order W
  For each material PO M in W
    if M has been shipped
      add returnCost(M) to penalty
```

Dealing with the cancellation business rule

To calculate the cancellation penalty, we need:

- Data about the status of the business process:
  - which instances of the “Fulfill work order” subprocess have already generated a material PO
  - which instances of the “Order Material” subprocess have been shipped, etc.

  Recorded by the PAIS (typically in its own database).

- Data about the suppliers to which each material order is sent:
  - the return policies of these suppliers
  - the costs for returning materials back to the suppliers

  Kept in different modules of the companys enterprise system.

Dealing with cancellation business rule (cont.'d)

Options for implementing a "Cancel Order" operation:

1. Insert additional tasks in the executable process model to pull the required data from the enterprise system:
   - Aggregated data and process execution status data passed to a "penalty calculation service".

   Drawback: introduce low-level data collection tasks in the executable process model.

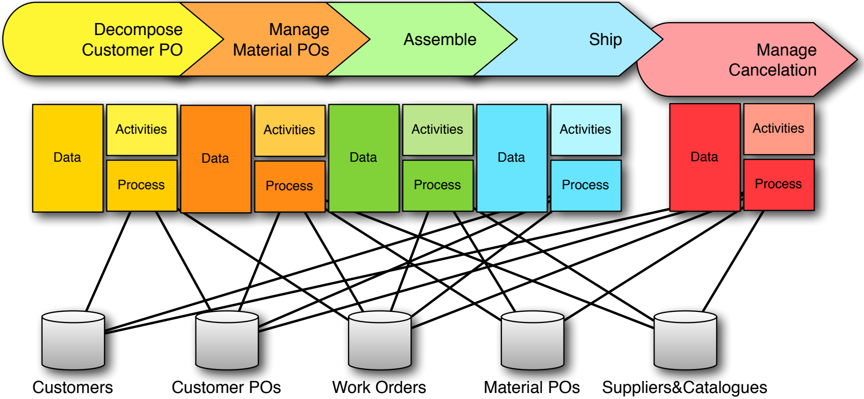
2. Push process execution status data from the PAIS to the enterprise system:
   - The PAIS invokes a "penalty calculation service" (implemented directly on top of the enterprise system) that fetches all the required data from the enterprise system and computes the penalty.

   Drawbacks:
   - process execution status data kept redundantly in the PAIS and in the enterprise system
   - additional tasks in the executable process model to update the enterprise system after each process step

Problem

The business process and the evolving data entities are conceived separately.

Consequence: Spaghetti



Overcoming the dichotomy

Strong need for:

- suitable modeling formalisms supporting the integrated management of processes and data;
- methodologies for the design of systems based on such formalisms;
- systems and tools that implement these languages and methodologies.

This, in turn requires a foundational approach to:

- provide a clear understanding of (data-aware) process models w.r.t.
  - semantic properties, and
  - computational properties;
- enable **analysis** of such formalisms.

Outline

1. Data and processes: two sides of the same coin
2. Data-centric processes
3. The Guard-Stage-Milestone meta-model
4. Data-Centric Dynamic Systems

Data-centric processes

Many real-world processes are:

- semi/loosely structured or unstructured
- knowledge-intensive
- driven by user decisions and data

Hence:

- Main driver for process progression: availability and status of data objects
- These processes cannot be represented as a set of activities with predefined precedence relations (control flow)

To model and implement data-driven processes, we need a tight integration of processes and data.

- Data must be modeled taking into account that they will be manipulated by processes.
- Processes must be modeled by considering that they are meant to manipulate data.

Data-centric process support

Data elements are considered as “first-class citizens” and as main drivers for process modeling and execution.

Data-centric approaches:

- Aim at providing a complete integration between the process and data perspectives.
- Rely on design methodologies where the identification and definition of process activities are induced by and follow the specification of the information and data model.

In data-centric methodologies:

- The data perspective is predominant:
  - Captures domain-relevant object types, their attributes, their possible states and life-cycles, and their interrelations, which together form a complex data structure or information model.
- The data model enables the identification and definition of the activities that operate on the object-related information, producing changes on attribute values, relations and object states.

Data-centric process management approaches

We discuss two of the main data-centric process management approaches:

- The artifact-centric paradigm [IBM Research et al.]
- Relational data-centric dynamic systems (DCDSs)
  - Pristine formalization of the artifact-centric approach.
  - Relational data layer: keeps all the data of interest and holds the relevant information to be manipulated by the system.
  - Process layer: invokable actions and a declarative rule-based process that modifies and evolves the data.

The artifact-centric approach

- In early 2000, the **artifact-centric approach** emerged as a foundational proposal for merging data and processes.
  - The emphasis is on data, which are modeled taking into account that they will be manipulated by processes.
  - Processes are conceived in terms of collections of artifacts that encapsulate data and have an associated lifecycle.
- Initial proposals by IBM [Nigam and Caswell 2003], followed by [Bhattacharya et al. 2007; Deutsch, Sui, and Vianu 2007], . . .
- See also EU project ACSI (for **Artifact-Centric Service Interoperation**), 2010–2013.

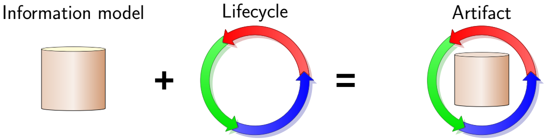
What is an artifact?

Artifact

A key, business-relevant conceptual dynamic entity that is used in guiding the operation of a business, and whose content evolves as it moves through those operations.

An artifact type consists of:

- **Information model** - relevant data maintained by the artifact
- **Lifecycle model** - (implicit) description of the allowed information model evolutions through the execution of a process.

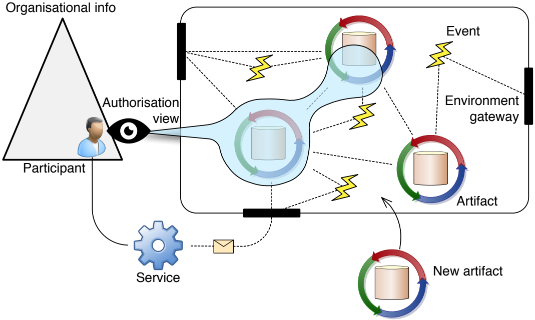


Goal

Provide unified, end-to-end view of relevant entities and their possible evolutions.

Artifact-based system

Is a set of interacting artifacts immersed in an environment.



- The environment represents the external world, including users, external data evolutions. sources, and services, in which the system runs.
- Artifacts interact with environment through events/messages with payload
  - Artifacts receive externally generated incoming events/messages from env.
  - Artifacts' progression produces outgoing events/messages towards env.

Artifacts and information model

Information model or **data schema** of a business artifact type

- Holds, in a hierarchical description, the information needed in completing business process execution in connection with a given business entity.
- Is defined as a set of attributes.
- Includes all data to capture business process goals and their achievement.

Artifact instance of a specific type

- Has a value for each attribute.
- Artifact instances can be created and destroyed over time.
- Attributes and their values can be created, updated, or deleted by the services in the process environment, according to their data schema.

The information contained in the set of identified artifacts records all the information about the status and progress of a process.

$\rightsquigarrow$ The runtime state of a business process is determined by the current snapshot of all artifacts.

Artifacts in the build-to-order process

The process now centers around interconnected business-relevant entities:

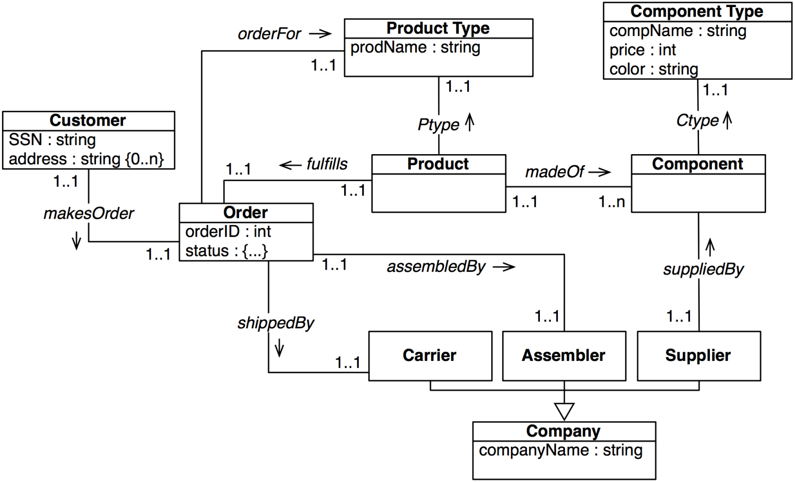
- Customer PO handles a customer order from creation to delivery.
- Work Order handles one of the work orders spawned for a line item in a customer PO.
- Material PO handles a material PO from request to shipment (and possible rejections).
- Assembly manages the aggregation of materials and sub-assemblies.

Open points:

- How to specify the lifecycle of such artifacts?
- At which level of abstraction?
- How and where to store data maintained by their information models?

Build-to-order – Information model

Conceptual schema of: Customer Purchase Order



unibz

Artifacts and lifecycles

The **lifecycle model** of a business artifact type identifies business-relevant phases in the possible evolution of artifact instances of that type.

- Describes how an artifact can evolve over time.
- Specifies possible ways in which an artifact might progress through the business.
- Defines the ways in which it will respond to events and invoke external services.

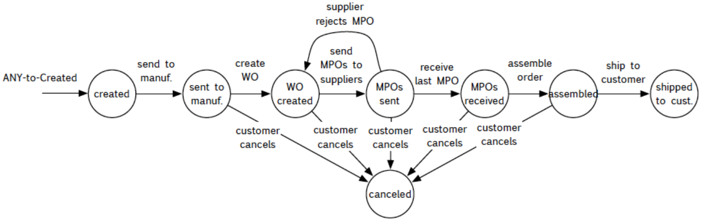
The artifact-centric paradigm does not restrict the way to specify artifact lifecycles:

- A lifecycle is generally defined by the specification of a set of **dynamic constraints** on the allowed sequencing of the phases traversed by artifact instances.
- Intra-artifact constraints relate artifact phases across time.

Concrete lifecycle models

The constraints defining the artifacts' lifecycles can be specified:

1. Using a procedural approach based on Finite State Machines (FSM)
   - states of the machine correspond to phases/stages in the lifecycle
   - guarded transitions control state changes
   - each transition is associated with a flow that models the sequencing of tasks that operate over the data and/or represent interactions with external services.
   - pro: easy to understand con: hard to specify and modify



2. Using a declarative rule-based approach:
   ⇝ Guard-Stage-Milestone (GSM) concrete meta-model

Outline

1 Data and processes: two sides of the same coin

2 Data-centric processes

3 The Guard-Stage-Milestone meta-model

4 Data-Centric Dynamic Systems

The Guard-Stage-Milestone (GSM) meta-model

GSM is a **declarative rule-based** framework for the specification of artifact-centric processes.

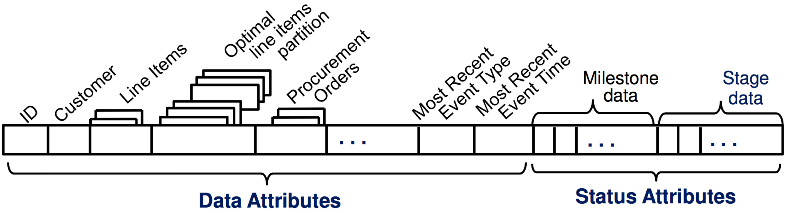
Consists of four main components:

1. The information model
2. Guards
3. Stages
4. Milestones

Information model

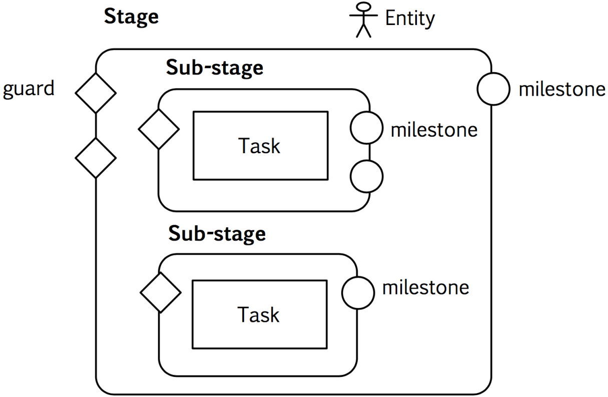
Provides an integrated view of all business-relevant information about artifact instances (as in the general paradigm).

- Relational schema (with nested records)
- Attributes partitioned into:
  - data attributes: hold business-relevant data about the progress of an artifact instance
  - status attributes: hold information about the current status of stages (open/closed) and milestones (achieved/invalidate) [see later]



GSM artifact lifecycles

Artifact life-cycles are defined in terms of stages, each associated with one or more milestones and one or more guards.



Stages, sub-stages, and tasks

A **stage** identifies a cluster of tasks and sub-stages related to an artifact instance. Stages can be:

- composite: support the nesting of (sub-)stages
- atomic: contain tasks that consist in the execution of specific activities or in the invocation of services in the environment

Tasks may result in:

- the assignment or modification of attribute values
- outgoing events/messages
  - the invocation of one-way or two-way external services (including user interactions as two-way service calls, i.e., human services)
  - the request to send a message as a response to an incoming service call
  - the request to send a message/event to other artifact instances
  - the request to create a new artifact instance

Task executions may produce output data that is written back into the artifact instance information model.

Guards and milestones

The activation or opening of stages is controlled by the associated guards.

- A **guard** is an expression (sentry, see later) that determines whether a stage becomes active (or open).
- A stage becomes active only if one of its guards becomes true and its parent stage (if any) is open.
- When an atomic stage becomes active, the corresponding task can be executed.

The closing of stages is controlled by the associated milestones.

- A **milestone** is an expression (sentry) that captures a business-relevant operational objective/goal (at different levels of granularity).
- A milestone:
  - can be achieved (and become true) by an artifact instance
  - may be invalidated (and become false)
- A stage is closed (or inactive) when one of its milestones is achieved (at most one milestone of a stage can be true), or when its parent stage closes.

Sentries for guards and milestones

Guards and milestones represent the key elements that determine the progress of artifact instances.

⤳ Are declaratively defined by expressions based on triggering events and/or conditions.

Expressions for guards and milestones are called **sentries** and have the form

$$\textbf{on}\ \langle \textit{event} \rangle\ \textbf{if}\ \langle \textit{condition} \rangle$$

where:

- The (optional) triggering event $\langle \textit{event} \rangle$ may correspond to
  - an incoming external event from the environment: 1-way message, service call return, artifact creation request
  - an internal event representing a status change of a stage (becomes open / closes) or a milestone (achieved / invalidated)
- The (optional) condition $\langle \textit{condition} \rangle$ defines a boolean formula over the artifact's information model.

From sentries to ECA-like rules

The set of sentries for an artifact lifecycle results in a family of Event-Condition-Action-like (ECA) rules that control the lifecycle of the artifact.
$\leadsto$ **Prerequisite-Antecedent-Consequent** (PAC) rules

For example, for each stage S, and for each guard

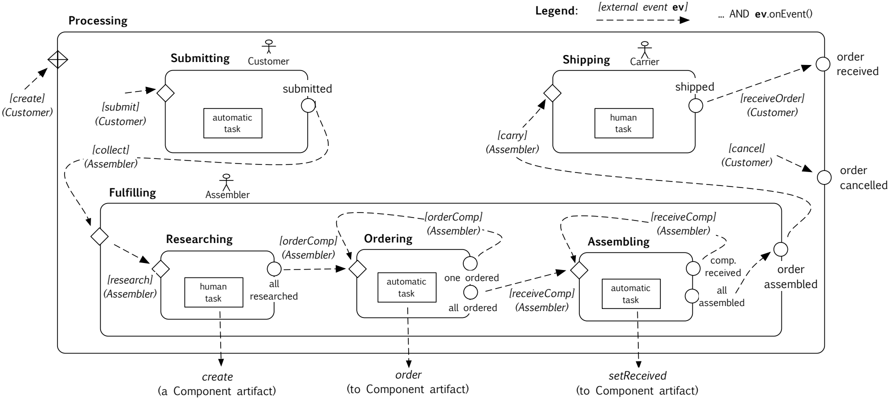
$$g\text{: } \textbf{on } \langle \textit{event} \rangle \textbf{ if } \langle \textit{condition} \rangle$$

of S, we have the following rule:

- Prerequisite: stage S is not active
- Antecedent: **on** ⟨*event*⟩ **if** ⟨*condition*⟩
- Consequent: stage S becomes active/open

The evaluation and sequencing of these rules determine the runtime evolution of an artifact.

Build-to-order process in GSM – Customer PO lifecycle



Sentries are partially hidden.
Arrows denote dependencies on certain status attributes.

Example: e-commerce

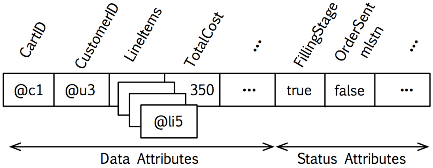
Process

Consider the process where a customer purchases products (called line items) from an e-commerce website.

Assuming that the customer is already logged in, the process consists of the following steps:

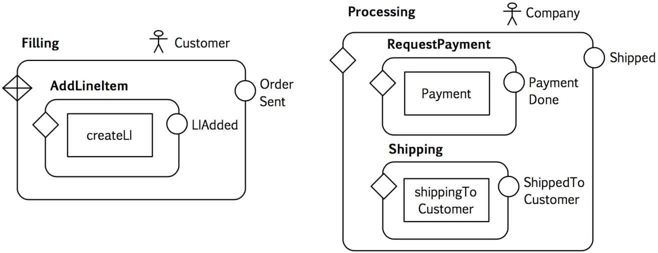
1. The customer adds one or more line items to the shopping cart.
2. The customer sends the order.
3. The company processes the payment and the shipping of the order.

Information model of the shopping cart artifact:



Example: e-commerce (cont.'d)

Shopping cart artifact lifecycle:



[g] *AddLineItemGuard*: **on** *chooseItem*;

[m] *LIAdded*: **on** *createLICompleted*;

[m] *OrderSent*: **on** *checkoutEvent* **if** *LineItems* $\neq \emptyset$;

[g] *ProcessingGuard* : **on** $+$*OrderSent*;

[g] *RequestPaymentGuard*: **on** $+$*OrderSent* **if** $\neg$*PaymentDone*;

[m] *PaymentDone*: **on** *PaymentCompleted*;

[g] *ShippingGuard*: **on** $+$*PaymentDone*;

[m] *ShippedToCustomer*: **on** *ShippingToCustomerCompleted*;

[m] *Shipped*: **on** $+$*ShippingToCustomer* **if** *PaymentDone*;

Event- and data-driven progression

Artifact instances move through their lifecycles as the result of events. When processed, they may result in:

- changes in the data, and
- a series of guards becoming true, and/or
- milestones changing value, and
- stages becoming open and/or closed.

The operational semantics of GSM is centered around the concept of business steps (**B-step**)

- A B-step intuitively corresponds to the smallest unit of business-relevant change that can occur to a GSM system.
- A B-step focuses on what happens when a single incoming event is processed:
  1. Incorporation of the event payload and its immediate effect on the artifact data.
  2. Triggering and evaluation of sentries for guards (to open stages and execute contained tasks) and milestones (achieving and invalidating sentries).

GSM B-step: Intuition

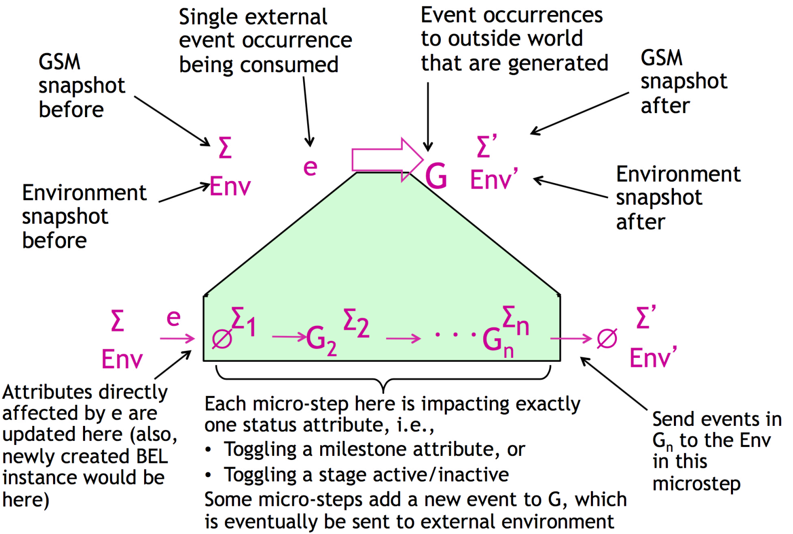
Given an incoming event from the environment (e.g., 1-way user-generated message):

- Immediate effect: event payload is used to change data attributes in the artifact.
- The event may trigger:
  - sentries for guards: stages may change status and become open/active
  - sentries for milestones: milestones may change status (achieved/invalidated) and stages may close
- The opening of atomic stages results in the execution of the enclosed tasks:
  - outgoing events/messages are generated towards the environment (e.g., service call)
- Status change of stages and milestones represent internal events
  - May in turn trigger other sentries for guards and milestones.

$\leadsto$ B-step as a sequence of “micro-steps”, each impacting exactly one status attribute for stages and milestones:

- toggling a stage active/inactive, or
- toggling a milestone achieved/invalidated

GSM B-step and micro-steps



GSM snapshot before

Single external event occurrence being consumed

Event occurrences to outside world that are generated

GSM snapshot after

Environment snapshot before

Environment snapshot after

Attributes directly affected by e are updated here (also, newly created BEL instance would be here)

Each micro-step here is impacting exactly one status attribute, i.e.,

- Toggling a milestone attribute, or
- Toggling a stage active/inactive

Some micro-steps add a new event to G, which is eventually be sent to external environment

Send events in G_n to the Env in this microstep

Formulations of the GSM operational semantics

GSM operational semantics is based on

- notion of B-step
- variation of Event-Condition-Action (ECA) rules derived from sentries

Three alternative, equivalent formulations for the operational semantics of GSM have been formally defined:

1. Incremental formulation: based on the incremental application of ECA-like rules (as intuitively discussed before)
2. Fixpoint formulation
3. Closed-form formulation

Guard-Stage-Milestone recap (1/2)

Information model: nested records

Lifecycle: constituted by

- Events, triggering the progression of the lifecycle
  - external events inject fresh data into the system
  - internal events mark changes of the lifecycle state
- Atomic tasks, used to create/destroy artifacts and to interact with the environment (users, services, etc.)
- Sentries, logical formulae combining an optional triggering event and a condition/query over the information model
- Milestones, business-relevant objectives that can be achieved by making the corresponding sentry true, invalidated otherwise
- Stages, clusters of tasks and sub-stages that jointly concur to the achievement of some milestones
- Guards, sentries used to control the activation of a corresponding stage:
  - Whenever a guard becomes true, the stage opens and must eventually reach a milestone to become closed.

Guard-Stage-Milestone recap (2/2)

- Data and information model:
  - explicit representation of domain-relevant objects/artifacts, their attributes and inter-relations
  - characterization of objects/artifacts evolution and behaviour in terms of lifecycles
- Data- and event-driven modeling and execution:
  - Data models enables the definition of activities
  - Activities and life-cycle transitions enabled by triggering events, constrained by conditions over data attributes/states (ECA-like rules).
  - Executed activities produce changes on attribute values, object/artifact relations and states.
- Explicit representation of goals/milestones
  - Goal achievement induced by event occurrence and changes in the information model.

The ArtiFact system

Open-source system that supports and implements the artifact-centric framework.

- Developed by IBM Research (initially in the ACSI project).
- Supports FSM and GSM variants.
- Provides graphical modeling environment, execution engine and runtime environment.
- Provides multi-user support with authorization and access control policies.
- Available at `http://sourceforge.net/projects/bizartifact/`

Activity-centric vs. data-centric methodology

Activity-centric

1. Identification and definition of the set of activities to be performed and the execution relationships between them.
   - ⇝ Predominance of the control-flow perspective.
2. Identification of data elements consumed, produced and exchanged during process executions.
   - ⇝ Definition of process- and activity-level input/output variables and data-based routing conditions.
3. Identification and definition of organizational models that describe the available resources and their capabilities.
   - ⇝ Activities are linked to the resources able to execute them, according to specific allocation policies and strategies.

Activity-centric vs. data-centric methodology (cont.'d)

Data-centric

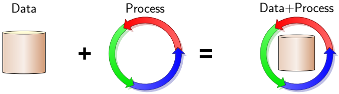
1. Identification and definition of the domain-relevant entities, their relationships and lifecycles.
    - Focus on the data perspective
    - Possibility to define relationships and dependencies between resources and data
        - Modeling of users and their roles in relation to data elements
        - Data-aware relationships between process activities and users
        - Authorization and access control policies, with data-aware constraints
2. Identification and definition of the activities that operate on the data entities.
    - Definition of data-aware execution guards/constraints (preconditions)
    - Definition of the impact of actions executions on the data (postconditions/effects)
    - Definition of data-aware allocation policies
3. Identification and definition of the overall process that drives and constraints action executions.
    - Explicit procedural specification or declarative specification with rules and constraints

Outline

1. Data and processes: two sides of the same coin
2. Data-centric processes
3. The Guard-Stage-Milestone meta-model
4. Data-Centric Dynamic Systems

Data-Centric Dynamic Systems (DCDS) [Bagheri Hariri et al. 2013]

- Abstract model underlying variants of artifact-centric systems.
- Semantically equivalent to the most expressive models for business process systems (e.g., GSM).



- Data Layer: Relational databases / ontologies
  - Data schema, specifying constraints on the allowed states
  - Data instance: state of the DCDS
- Process Layer: key elements are
  - Atomic actions
  - Condition-action-rules for application of actions
  - **Service calls:** communication with external environment, **new data!**

An example: Hotels and price conversion

Data Layer: Info about hotels and room prices

$$\mathsf{Cur} = \langle \mathit{UserCurrency} \rangle \quad \mathsf{CH} = \langle \underline{\mathit{Hotel}}, \mathit{Currency} \rangle \quad \mathsf{PEntry} = \langle \underline{\mathit{Hotel}}, \mathit{Price}, \underline{\mathit{Date}} \rangle$$

Process Layer/1

User selection of a currency.

- Process: $\mathit{true} \longmapsto \mathsf{ChooseCur}()$
- Service call for currency selection: $\textsc{uInputCurr}()$
  - Models user input with **non-deterministic** behavior.
- $\mathsf{ChooseCur}() : \left\{ \begin{array}{r} \mathsf{Cur}(c) \rightsquigarrow \textbf{del}\{\mathsf{Cur}(c)\} \\ \mathit{true} \rightsquigarrow \textbf{add}\{\mathsf{Cur}(\textsc{uInputCurr}())\} \end{array} \right\}$

An example: Hotels and price conversion

Data Layer: Info about hotels and room prices

$\mathsf{Cur} = \langle \mathit{UserCurrency} \rangle \quad \mathsf{CH} = \langle \underline{\mathit{Hotel}}, \mathit{Currency} \rangle \quad \mathsf{PEntry} = \langle \underline{\mathit{Hotel}}, \mathit{Price}, \underline{\mathit{Date}} \rangle$

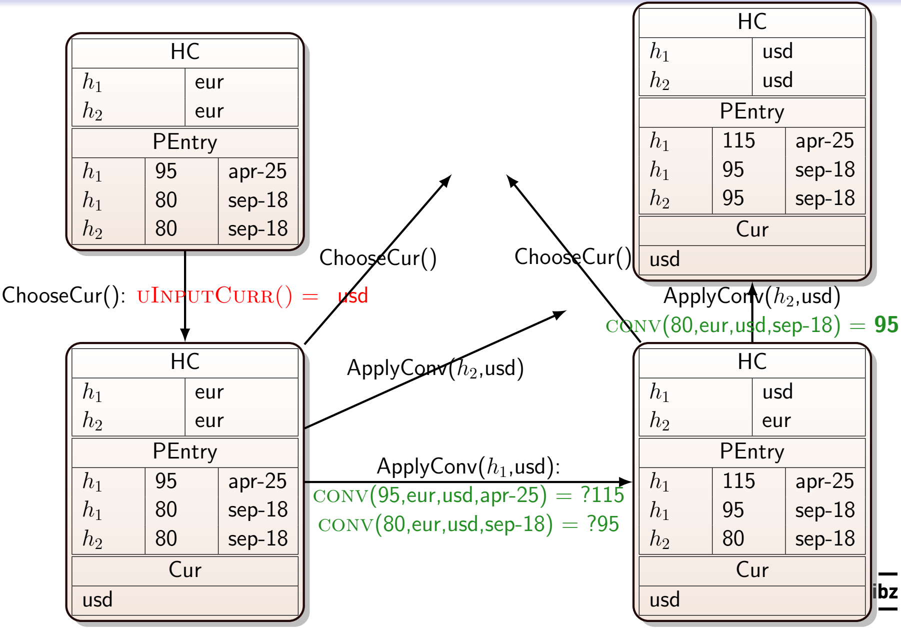
Process Layer/2

Price conversion for a hotel.

- Process: $\mathsf{Cur}(c) \land \mathsf{CH}(h, c_h) \land c_h \neq c \longmapsto \mathsf{ApplyConv}(h, c)$
- Service call for currency selection: $\textsc{conv}(\mathit{price}, \mathit{from}, \mathit{to}, \mathit{date})$
  - Models historical conversion with deterministic behavior.
- $\mathsf{ApplyConv}(h, c)$:

$$\left\{ \begin{array}{r} \mathsf{PEntry}(h,p,d) \rightsquigarrow \mathbf{del}\{\mathsf{PEntry}(h,p,d)\} \\ \mathsf{PEntry}(h,p,d) \land \mathsf{CH}(h,c_{old}) \land \mathsf{Cur}(c) \rightsquigarrow \mathbf{add}\{\mathsf{PEntry}(h, \textsc{conv}(p, c_{old}, c, d), d)\} \\ \mathsf{CH}(h,c_{old}) \rightsquigarrow \mathbf{del}\{\mathsf{CH}(h,c_{old})\},\ \mathbf{add}\{\mathsf{CH}(h,c)\} \end{array} \right\}$$

Run of the system



HC	
h_1	eur
h_2	eur

PEntry		
h_1	95	apr-25
h_1	80	sep-18
h_2	80	sep-18

ChooseCur(): $\textsc{uInputCurr}() =$ usd

HC	
h_1	eur
h_2	eur

PEntry		
h_1	95	apr-25
h_1	80	sep-18
h_2	80	sep-18

Cur
usd

ChooseCur()

ApplyConv(h_2,usd)

ApplyConv(h_1,usd):

$\textsc{conv}$(95,eur,usd,apr-25) $=$?115

$\textsc{conv}$(80,eur,usd,sep-18) $=$?95

HC	
h_1	usd
h_2	eur

PEntry		
h_1	115	apr-25
h_1	95	sep-18
h_2	80	sep-18

Cur
usd

ChooseCur()

ApplyConv(h_2,usd)

$\textsc{conv}$(80,eur,usd,sep-18) $=$ **95**

HC	
h_1	usd
h_2	usd

PEntry		
h_1	115	apr-25
h_1	95	sep-18
h_2	95	sep-18

Cur
usd

Deterministic vs. non-deterministic services

DCDSs admit two different semantics for service-execution:

Deterministic services semantics

Along a run, when the same service is called again with the same arguments, it returns the same result as in the previous call.

Are used to model an environment whose behavior is completely determined by the parameters.

Example: temperature, given the location and the date and time

Non-deterministic services semantics

Along a run, when the same service is called again with the same arguments, it may return a different value than in the previous call.

Are used to model:

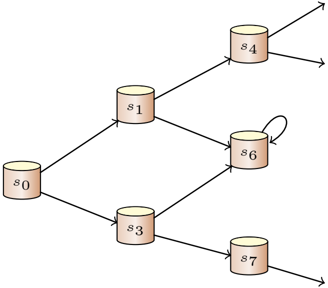
- an environment whose behavior is determined by parameters that are outside the control of the system;
- input of external users, whose choices depend on external factors.

Example: current temperature, given the location

Semantics via transition systems

Semantics of a DCDS $\mathcal{S}$ is given in terms of a **transition system** $\Upsilon_{\mathcal{S}}$:

- each state of $\Upsilon_{\mathcal{S}}$ has an associated DB over a common schema;
- the initial state is associated to the initial DB of the DCDS.



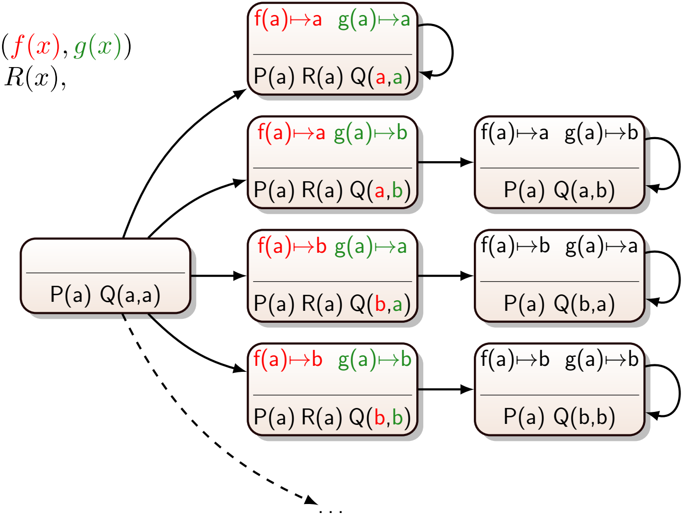
Note: $\Upsilon_{\mathcal{S}}$ is in general **infinite state**:

- infinite branching, due to the results of service calls,
- infinite runs, since infinitely many DBs may occur along a run;
- the DBs associated to the states are of unbounded size.

Transition system – Example for deterministic services

$$\left\{ \begin{array}{l} P(x) \rightsquigarrow P(x) \wedge Q(f(x), g(x)) \\ Q(a,a) \wedge P(x) \rightsquigarrow R(x), \end{array} \right.$$

$$\mathcal{I} = \{P(a), Q(a,a)\}$$



Verification for DCDSs

We are interested in the **verification** of temporal properties over $\Upsilon_{\mathcal{S}}$.

Idea to overcome infiniteness:

1. Devise a **finite-state** transition system $\Theta_{\mathcal{S}}$ that is a **faithful abstraction** of $\Upsilon_{\mathcal{S}}$ **independent of the formula** to verify.
2. Reduce the verification problem $\Upsilon_{\mathcal{S}} \models \Phi$ to the verification of $\Theta_{\mathcal{S}} \models \Phi$.

Problem: Verification of DCDSs is undecidable even for propositional reachability properties.
$\leadsto$ We need to pose restrictions on DCDSs.

This aspect will be dealt with in Part 3 of the course.

Acknowledgements

Thanks to the many people who contributed interesting ideas, suggestions, discussions, collaborated to the presented results, and provided material at used for the presentation slides.

Giuseppe De Giacomo
Marco Montali

Babak Bagheri Hariri
Riccardo De Masellis
Alin Deutsch
Paolo Felli
Rick Hull
Maurizio Lenzerini
Alessio Lomuscio
Fabio Patrizi
Alessandro Russo
Ario Santoso
Moshe Vardi

Thank you for your attention!

References I

[1] Rob Karel, Clay Richardson, and Connie Moore. *Warning: Don't Assume Your Business Processes Use Master Data – Synchronize Your Business Process And Master Data Strategies*. Report. Forrester, Sept. 2009.

[2] Manfred Reichert. "Process and Data: Two Sides of the Same Coin?". In: *Proc. of the On the Move Confederated Int. Conf. (OTM 2012)*. Vol. 7565. Lecture Notes in Computer Science. Springer, 2012, pp. 2–19. DOI: http://dx.doi.org/10.1007/978-3-642-33606-5_2.

[3] Andreas Meyer, Sergey Smirnov, and Mathias Weske. *Data in Business Processes*. Tech. rep. 50. Available online at http://opus.kobv.de/ubp/volltexte/2011/5304/. Hasso-Plattner-Institut for IT Systems Engineering, Universität Potsdam, 2011.

[4] Anil Nigam and Nathan S. Caswell. "Business Artifacts: An Approach to Operational Specification". In: *IBM Systems Journal* 42.3 (2003), pp. 428–445. DOI: http://dx.doi.org/10.1147/sj.423.0428.

References II

[5] Kamal Bhattacharya et al. "Towards Formal Analysis of Artifact-Centric Business Process Models". In: *Proc. of the 5th Int. Conf. on Business Process Management (BPM)*. Vol. 4714. Lecture Notes in Computer Science. Springer, 2007, pp. 288–234. DOI: http://dx.doi.org/10.1007/978-3-540-75183-0_21.

[6] Alin Deutsch, Liying Sui, and Victor Vianu. "Specification and Verification of Data-Driven Web Applications". In: *J. of Computer and System Sciences* 73.3 (2007), pp. 442–474.

[7] Babak Bagheri Hariri et al. "Verification of Relational Data-Centric Dynamic Systems with External Services". In: *Proc. of the 32nd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS)*. 2013, pp. 163–174.