
SAPIENZA
UNIVERSITÀ DI ROMA

# The Guard-Stage-Milestone (GSM) Framework for Modeling Artifact-based Workflows

R. De Masellis, P. Felli

Sapienza Università di Roma
Dipartimento di Informatica e Sistemistica

Elective in Software and Services
2010-2011

# What is an *Artifact*?

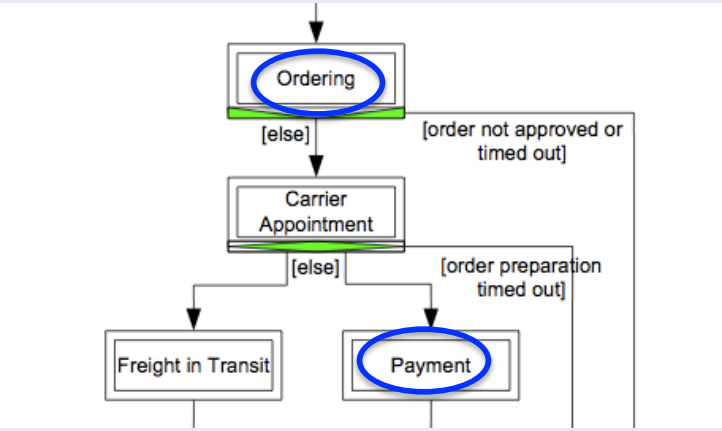
## Definition (Artifact)

*"It is a data record, evolving over time, representing a business relevant entity"*

Introduced by IBM as the core building block for workflows.

## **Activity-Centric** approach

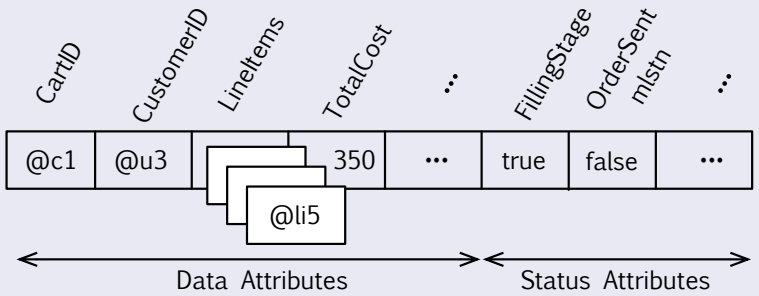
Focuses on processes:




## **Artifact-Centric** approach

Focuses on data as well as processes:

Shopping Cart Artifact:



# E-commerce Example

## Example (E-commerce)

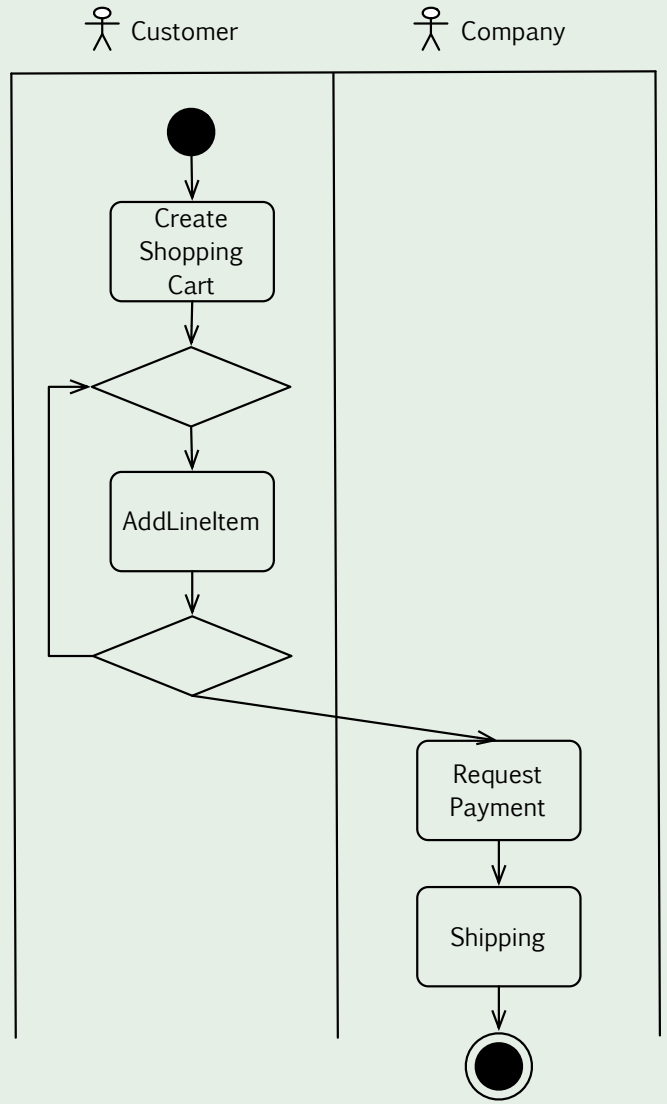
Consider the process where a *customer* purchases products (called *line items*) from an e-commerce website.

Assuming that the customer is already logged in, the process is composed by the following steps:

- the customer adds one or more *line items* to the *shopping cart*;
- the customer sends the order;
- the company processes the payment and the shipping of the order.

# Activity-Centric E-commerce Example

## Example (E-commerce process modeled with UML activity diagram)




**Question:**

Where are data?

# Activity-Centric E-commerce Example

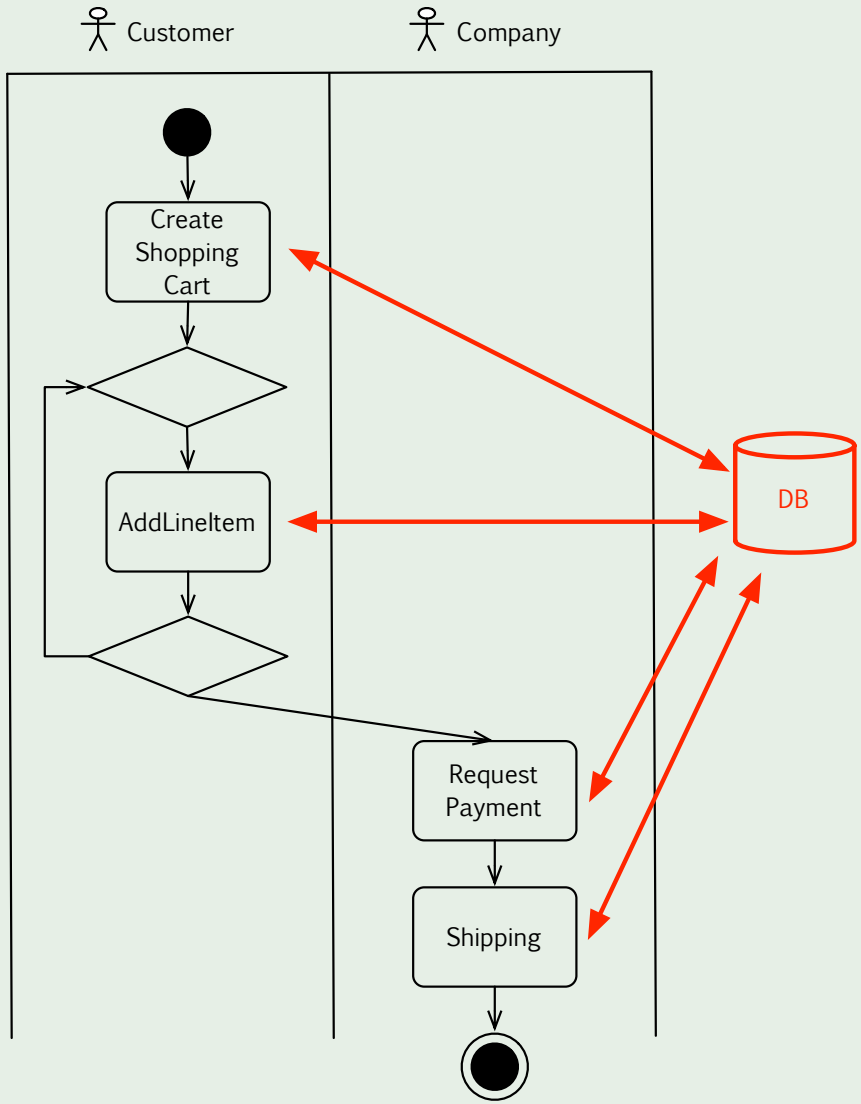
## Example (E-commerce process modeled with UML activity diagram)

**Question:**

Where are data?

**Answer:**

Outside the model
(not explicitly modeled)!



# Artifacts

**Definition (Artifact type)**

An artifact type is a tuple $A = \langle D, L \rangle$ where:

- $D$ is called data schema and it captures the artifact's relevant information;
- $L$ is the lifecycle which specifies the *evolution* of the artifact.

# Artifacts: Information Model

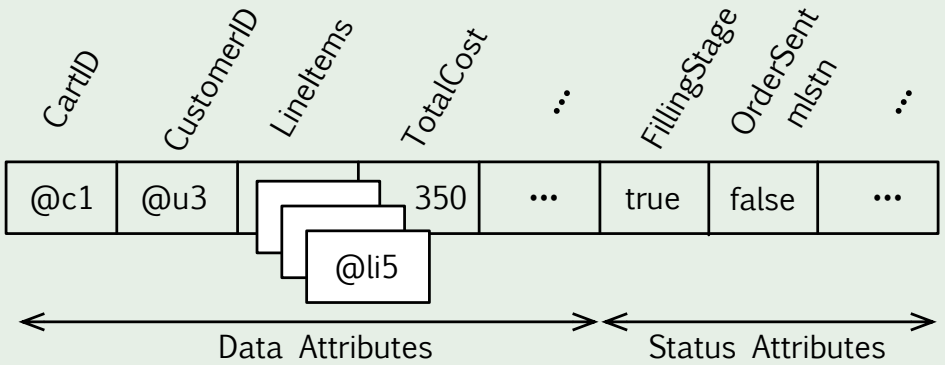
## Definition (Information Model)

An information model for artifact type $A$ is a set of *attributes*.

An artifact instance of type $A$ will have a value for each attribute.

## Example: Information model of the **shopping cart** artifact

$\{ \langle CartId, @c1 \rangle, \langle CustomerID, @u3 \rangle, \langle LineItems, \{@li1, @li2, @li3, @li5\} \rangle, \langle TotalCost, 350 \rangle, \ldots \}$



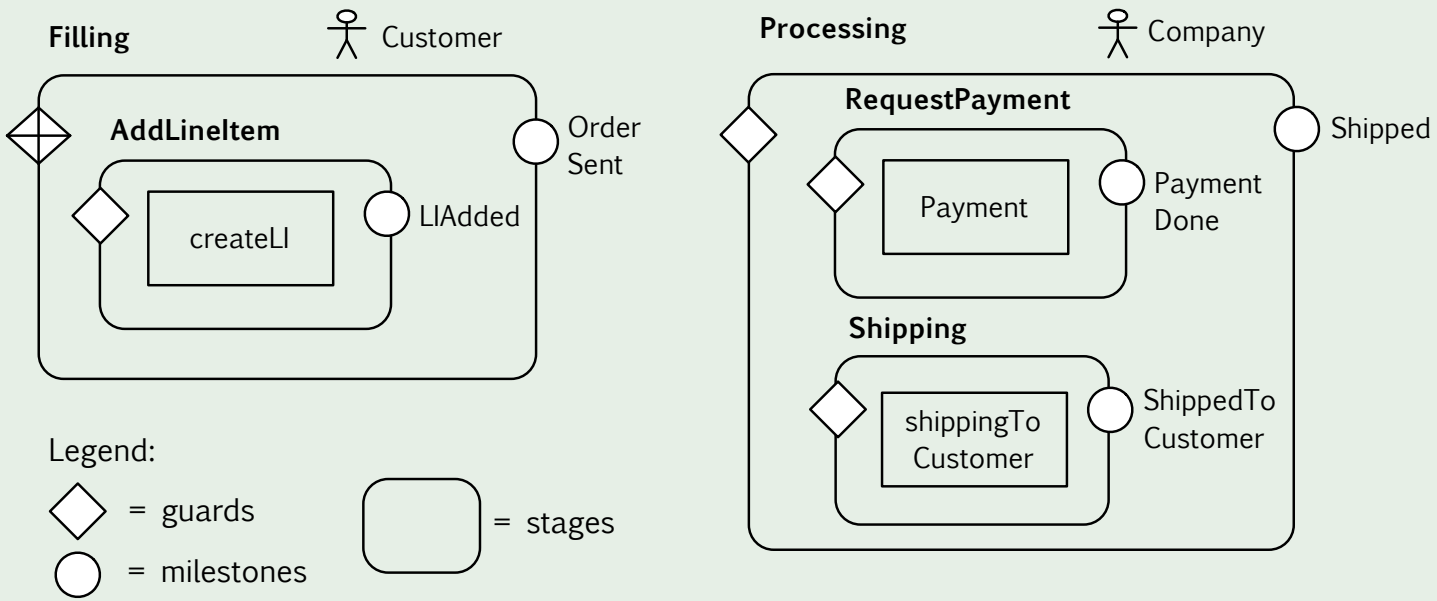
# Artifacts: Lifecycle

## Definition (Lifecycle)

A lifecycle for an artifact type $A$ is a set of triples $\langle G, S, M \rangle$ where:

- $G$ is a formula called guard;
- $S$ is called stage and it may be *atomic* or *composite*. If atomic, it contains a *task*, if composite, it is a set of $\langle G, S, M \rangle$;
- $M$ is set of formulas called milestones.

## Example: lifecycle of the **shopping cart** artifact



# Artifacts: Lifecycle

- Stages represent either atomic task or set of tasks organized in an implicit control flow;
- Guards control the *activation* of tasks: when the guard $g$ of a stage $s$ becomes true, then an activation of task(s) of $s$ starts (and we say that $s$ is "open");
- Milestones represents goals to achieve. When a milestone $m$ of a stage $s$ is achieved, $s$ "closes", i.e., the task(s) inside it cannot be performed anymore.

# Tasks

## Definition (Tasks)

A task is an activity specified in terms of *preconditions* and *postconditions*.

## Example (Specification of *CreateLI* task)

*CreateLI*(): void
*pre*: $\top$
*post*: a new instance *li* of line item artifact type is created and it is added to *this.LineItems*.

# Alphabet of Formulas

We said that *guards* and *milestones* are formulas, but over what?

**Definition (Alphabet of formulas)**

The *alphabet of formulas* for an artifact $A$ is composed by:

- information model's *attributes* of $A$;
- events.

**Example: a milestone formula**

$$OrderSent : LineItems \neq \emptyset \wedge checkoutEvent.onEvent()$$

# Events

We have two types of events:

- Externally generated events: events coming from the *environment*;
- Internally generated events:
  - events towards the *environment*;
  - milestone achieved, stage opens/closes, etc.

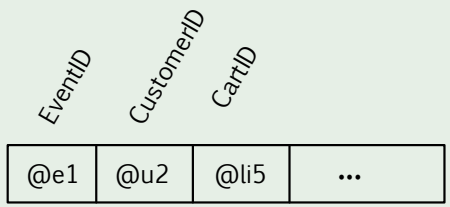
**Definition (Event Type)**

An event type $E$ is a set of *attributes*.

An event instance of type $E$ will have a value for each attribute.

**Example: an instance of event type *checkoutEvent*:**

$\{ \langle EventID, @e1 \rangle, \langle CustomerID, @u2 \rangle, \langle CartID, @c3 \rangle, \ldots \}$




**Remark**

Events, differently from informations models, are immutable.

# Environment

The environment represents the *external world*, namely everything that somehow interacts (through *external* events) with artifacts, like human users and external services (e.g., google maps).

Artifacts communicate with the environment through environment gateways and they receive externally generated event from them.

**Definition (Environment Gateway Type)**

An environment gateway type $V$ is a set of *attributes*.

An environment gateway instance of type $E$ will have a value for each attribute.

**Remark**

Data of externally generated event instances are taken from environment gateway instances.

# Environment

**Example (Example)**

In our e-commerce example, we have two environment gateway types: the *customer* and the *company*.

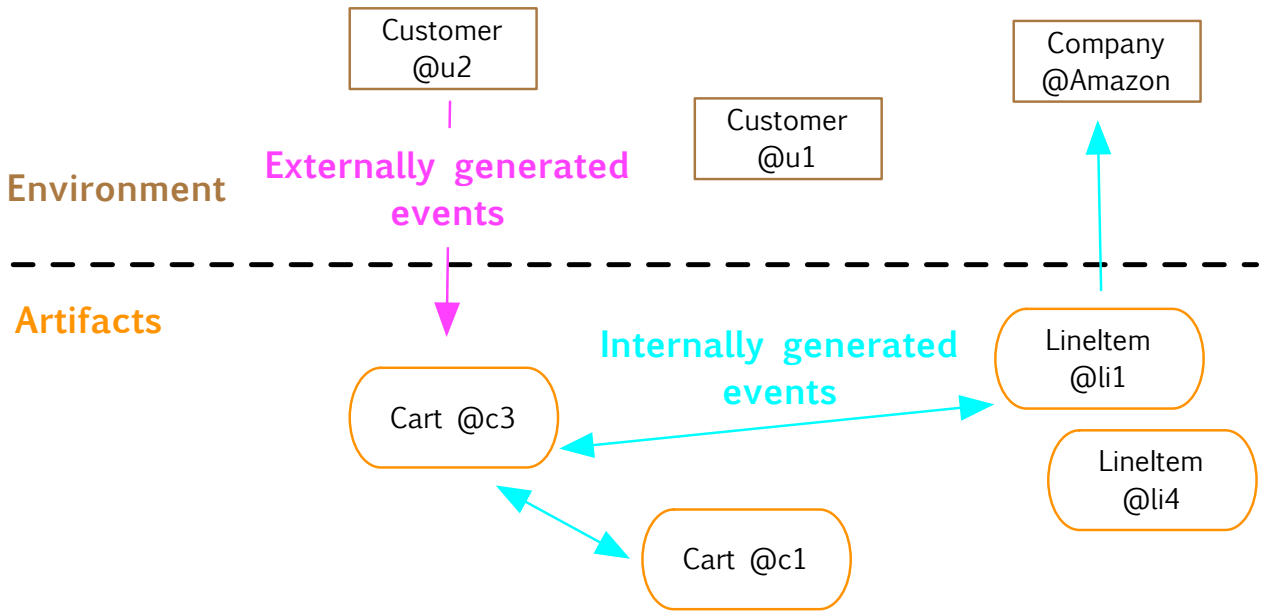
**Remark**

We cannot neither directly modify data of environment gateways nor control events generated by them.

# Recap

The framework's building blocks are:

- artifact types;
- environment gateway types;
- externally generated event types;
- internally generated events;



# Artifact-based System

**Definition (Artifact-based System)**

An artifact base system is a tuple $\mathcal{M} = \langle \mathcal{A}, \mathcal{V}, \mathcal{E} \rangle$ where:

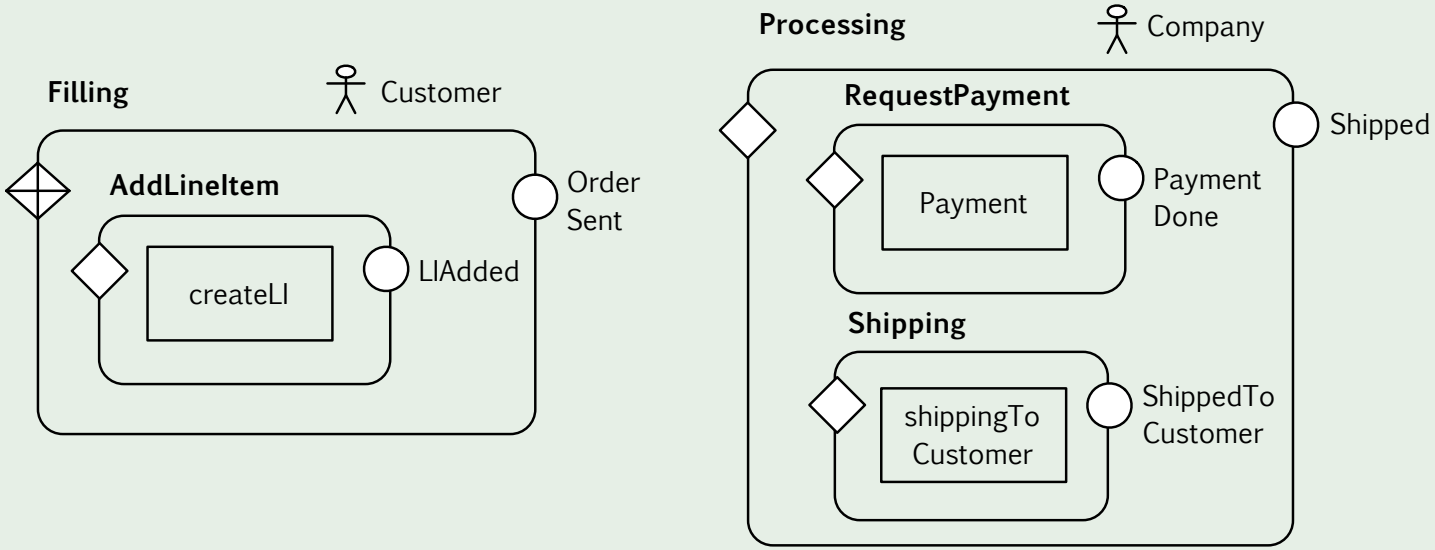
- $\mathcal{A}$ is a set of artifact types;
- $\mathcal{V}$ is a set of environment gateway types;
- $\mathcal{E}$ is a set of event types.

# The E-commerce Example

## Example (**Shopping Cart artifact** type)

**Information Model:** $\{cartID, CustomerID, LineItems, TotalCost\}$

**Lifecycle:**




$AddLineItemGuard : chooseItem.onEvent(); \ LIAdded : createLI.completed();$
$OrderSent : LineItems \neq \emptyset \wedge checkoutEvent.onEvent();$
$ProcessingGuard : OrderSent.achieved();$
$RequestPaymentGuard : OrderSent.achieved() \wedge \neg PaymentDone;$
$PaymentDone : Payment.completed()$
$ShippingGuard : PaymentDone.achieved();$
$ShippedToCustomer : ShippingToCustomer.completed();$
$Shipped : ShippingToCustomer.achieved() \wedge PaymentDone.$

# The E-commerce Example

**Example (Enviroment gateway and event types)**

| | |
|---|---|
| **Environment gateways types:** | $Customer = \{CustomerID, preferences\}$; $Company = \{CompanyID\}$. |
| **External events types:** | $ChooseItem = \{CustomerID, cartID, ItemID\}$; $Checkout = \{CustomerID, cartID\}$. |

# Semantics of the System

The semantics of the system specify how it evolves.

## Definition (Snapshot)

A snapshot $s$ of an artifact-based system $M$ is the state in which $M$ is at a given time, and it is made up by all relevant information for $M$, precisely:

- artifact instances, with data and info about lifecycles (open stages, milestone achieved, etc.);
- the event just consumed;
- the events' queue (both internally and externally generated).

# Evolution of the System

From the initial snapshot:

1 **do forever**:
2  pick and consume an externally generated event $ext_i$;
4  **while** the internally generated events' queue is not empty:
5  pick and consume an internal event $int_j$;

$$s_0 \xrightarrow{ext_1} s_1 \xrightarrow{int_1} s_3 \xrightarrow{int_2} \ldots \xrightarrow{int_n} s_n \xrightarrow{ext_2} s_{n+1} \xrightarrow{int_{n+2}} \ldots$$

represent a possible system run.

# Evolution of the System

## Example (E-commerce possible run)

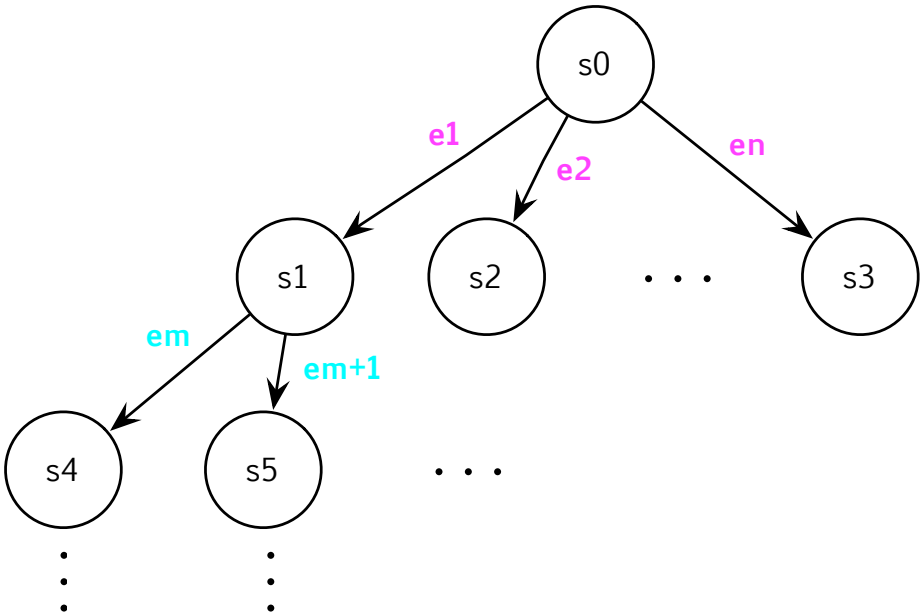
Suppose we have an artifact instance $c$ of *ShoppingCart* type and one gateway type instance $u$ of type *Customer*.

1. The event $e_1$ of type *ChooseItem* is picked and its consumption results in switching the *AddLineItemGuard* to true, opening stage *addLineItem* and starting the task *createLI*;
2. an internally generated event $e_2$ is produced as the task *createLI* completes;
3. the event $e_2$ is picked and its consumption results in achieving the *LIAdded* milestone, that in turn, produces an event $e_3$ of *milestone achievement*;
4. the event $e_3$ is picked and its consumption produces no effect (since there are no formulas with *LIAdded.achieved*()).
5. another externally generated event is picked ...
6. ...

# Evolution of the System

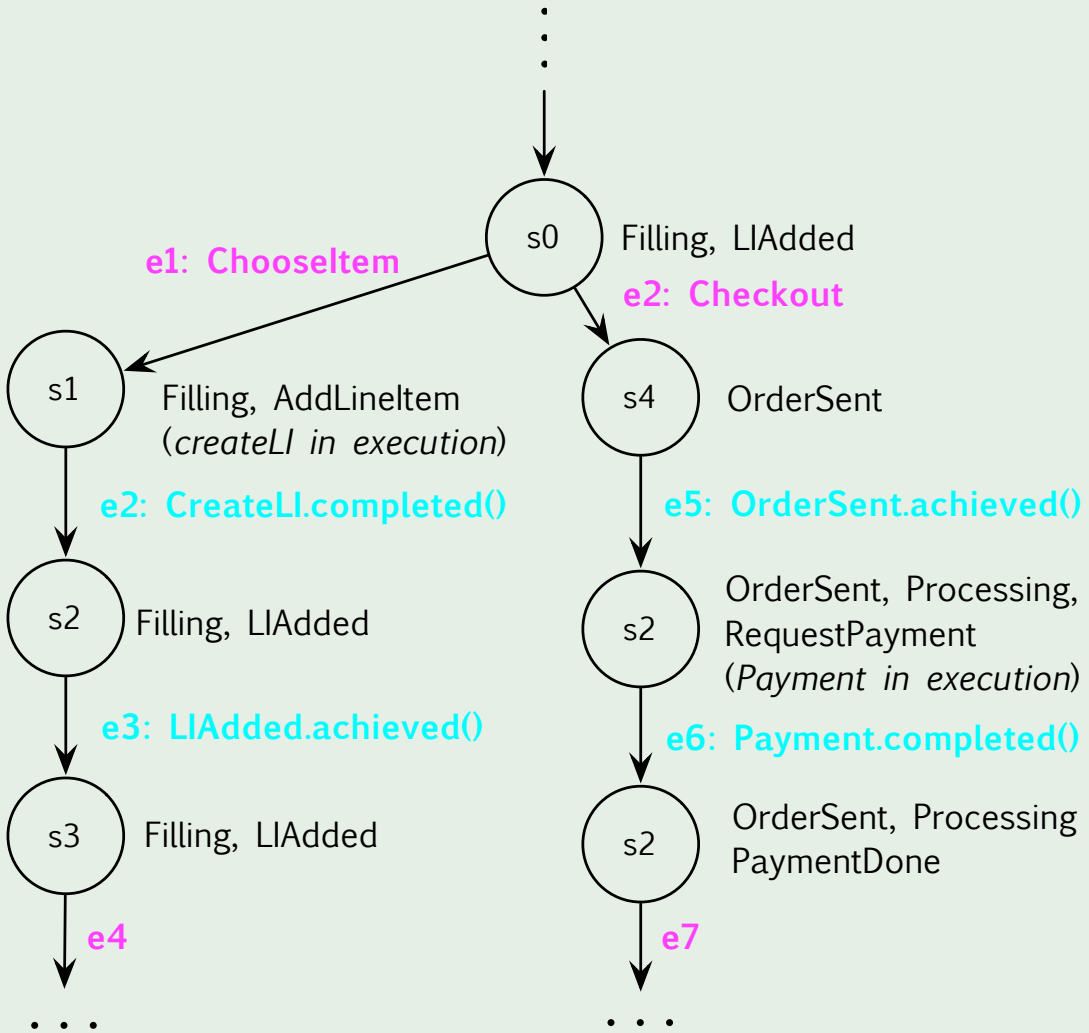
Every run depends on which events we choose...

Given an initial snapshot $s_0$, we can build a transition system representing all possible evolutions of the system!



# Evolution of the System

## Example (E-commerce all runs)



# Conclusions

We build the transition system in order to *explore* it,
for verification purposes.

**Bad news**

The transition systems has (in general) *infinite* states!

We cannot rely on "standard" model checking.

**Good news**

There are techniques [4, 3, 2, 1] that deal with this issue achieving interesting verification decidability results.

# Bibliography I


[1] G. D. G. Babak Bagheri Hariri, Diego Calvanese, R. D. Masellis, and P. Felli.

Foundations of relational artifacts verification.

In *BPM*, 2011.

[2] P. Cangialosi, G. D. Giacomo, R. D. Masellis, and R. Rosati.

Conjunctive artifact-centric services.

In *ICSOC*, pages 318–333, 2010.

[3] E. Damaggio, A. Deutsch, and V. Vianu.

Artifact systems with data dependencies and arithmetic.

In *ICDT*, pages 66–77, 2011.

[4] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu.

Automatic verification of data-centric business processes.

In *ICDT*, pages 252–267, 2009.

# Bibliography II


[5] R. Hull, E. Damaggio, F. Fournier, M. Gupta, F. T. Heath, S. Hobson, M. H. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculín.

Introducing the guard-stage-milestone approach for specifying business entity lifecycles.

In *WS-FM*, pages 1–24, 2010.

[6] R. Hull, N. C. Narendra, and A. Nigam.

Facilitating workflow interoperation using artifact-centric hubs.

In *ICSOC/ServiceWave*, pages 1–18, 2009.

[7] R. D. M. Richard Hull, Elio Damaggio, F. Fournier, M. Gupta, F. H. III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculin.

Business entities with guard-stage-milestone lifecycles: Managing entity interactions with conditions and events.

In *DEBS*, 2011.