Process Ontologies for Smart Objects in Manufacturing

Michael Grüninger

Track C: Decision Making in Different Domains
Ontology Summit 2015

February 9, 2015

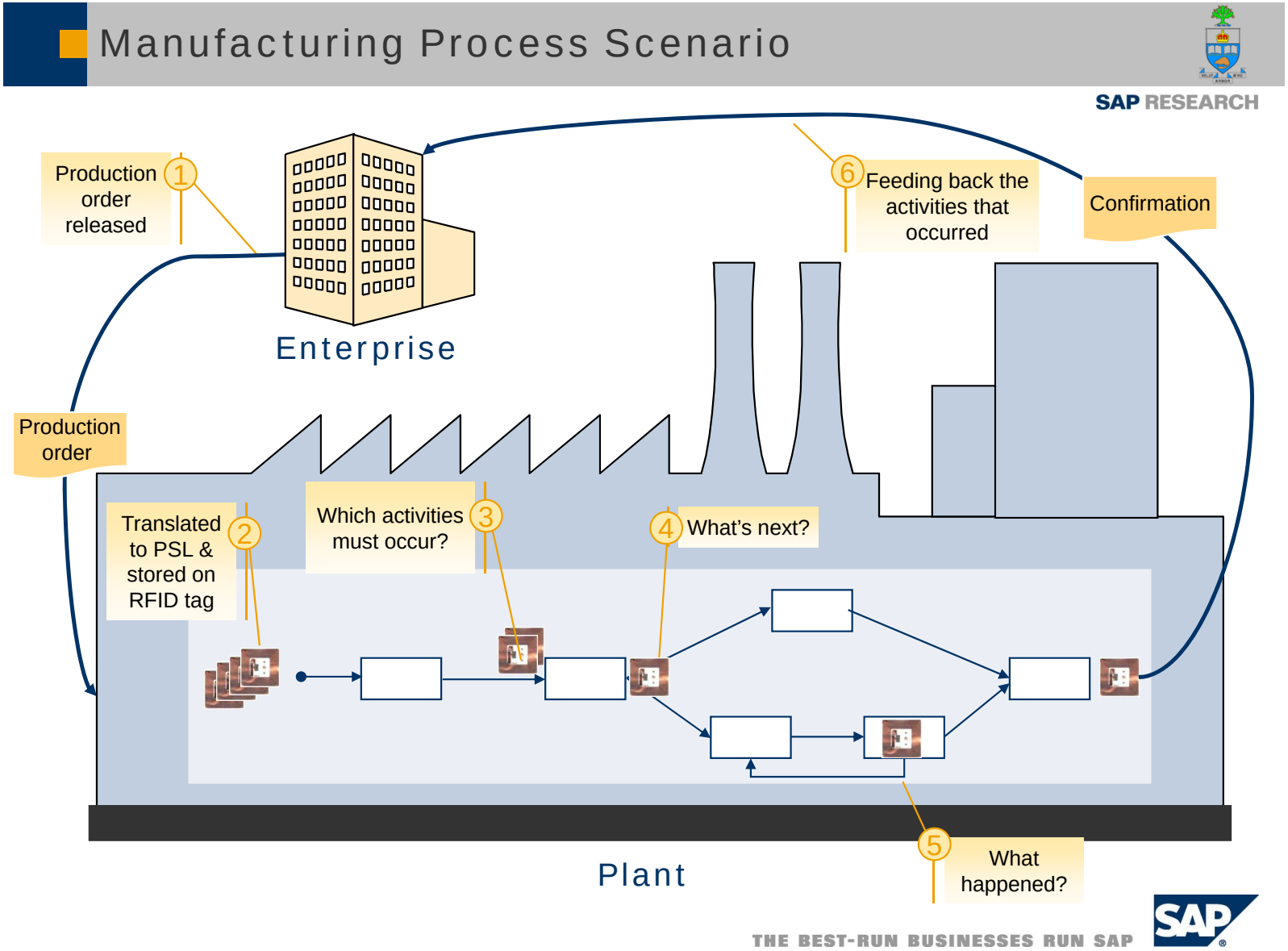
Scenario

- We are interested in dynamic self-routing of objects through the various process plans within the set of manufacturing processes.
- Each product is associated with a set of process plans, which are partially ordered sequences of manufacturing processes.
- In more general scenarios, such process plans may also be nondeterministic (that is, involve different choices of sequences of manufacturing processes).
- Objects "flow" through the sequence of processes. At any point in a process plan, there are multiple activities that can possibly occur next.
- Furthermore, different process plans may have manufacturing processes in common, so that an object may participate in an activity that is part of multiple process plans.

Manufacturing Process Control

- RFID technology has long been known for its ability to uniquely identify objects. As the tag itself can carry relevant context information, processes can be managed locally rather than relying on a centralized system infrastructure.
- Combine RFID technology with ontologies to create *smart items* and demonstrate this approach using a motivating scenario of manufacturing process control.
- Use a process and time ontology to store information directly on the RFID tags.
- As an item flows through a manufacturing process, information about the item can be stored on its tag. This information, along with the ontology's axioms can be used to make inferences about the manufacturing process and the item in particular.

Process Routing



Process Routing

- If something has gone wrong and there are quality problems, determine what the next activity can possibly occur (e.g. reroute in different process plan, rework within same process plan, scrap)
- Given the set of activities that have already occurred with the object, recognize which process plans can possibly be occurring. This is equivalent to identifying what the object can possibly become.
- Besides using only the ordering constraints in the set of process plans, this may also incorporate state constraints (properties of the object), temporal constraints (properties of the current schedule) and goal constraints (i.e. the set of current orders to be fulfilled).

Production Orders

- A production order can contain multiple "sequences" of operations. The sequences can be of three types: simple sequence, alternate sequence or parallel sequence.
- A simple sequence is a finite, linearly ordered sequence of operations. Each production order has a main simple sequence called the *standard sequence*.
- Branching off from the standard sequence can be parallel or alternate sequences. Each parallel or alternate sequence is also a simple sequence, but it also has start and end branch points to indicate the subsequence of the main sequence that it is parallel to or alternates with.
- Parallel sequences occur in parallel with the corresponding standard subsequence, but alternate sequences, if they are chosen for execution, will occur instead of the corresponding subsequence of the main sequence.

Informal Queries

- What activity can possibly occur next?
- What activities must occur in the production order?
- Does some activity a_1 always occur before some activity a_2 in the production order?

Why Use Ontologies?

This is the representation of the production order:

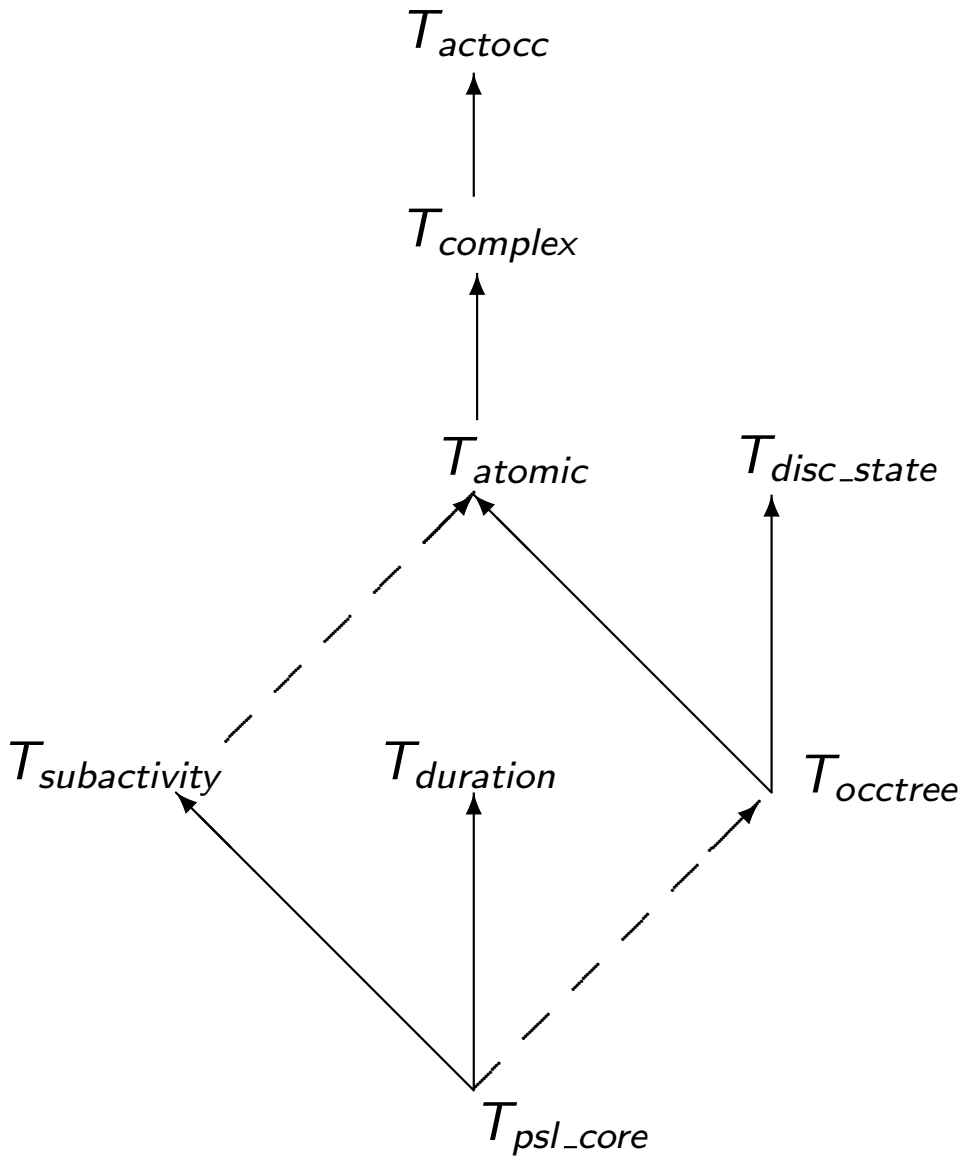
```
( E1AFKOL ( AUFNR "60004907" )( APRIO  )( APROZ "0.00" )( AUART "PP01" )
( AUFLD "20070831" )( AUTYP "10" )( BAUMNG "0.000" )( BMEINS "PCE" )
( BMENGE "10.000" )( CY_SEQNR "00000000000000" )( DISPO "101" )( FEVOR "101
( FHORI "001" )( FLG_MLTPS  )( FREIZ "005" )( FTRMI "20070815" )
( FTRMS "20070903" )( GAMNG "10.000" )( GASMG "0.000" )( GETRI "00000000" )
( GEUZI "000000" )( GLTRI "00000000" )( GLTRP "20070914" )
( GLTRS "20070912" )( GLUZP "000000" )( GLUZS "150000" )
( GMEIN "PCE" )( GSTRI "00000000" )( GSTRP "20070906" )( GSTRS "20070910" )
```

- The semantics are implicit in the algorithms for interpreting this data structure.
- We need a new algorithm for each new query.
- We cannot exchange algorithms or English documentation.

Process Specification Language

- PSL (ISO 18629) is a modular, extensible ontology capturing concepts required for process specification
- There are 300 concepts across 50 extensions of a common core theory (PSL-Core), each with a set of first-order axioms written using the Common Logic Interchange Format (ISO 24707).

Overview of PSL



Process Descriptions

Within the process plan for widgets, the resource is cut after fabrication.

```
(forall (?occ ?x)
        (if       (occurrence_of ?occ (make_widget ?x))
                  (exists (?occ1 ?occ2)
                        (and   (occurrence_of ?occ1 (fabricate ?x))
                               (occurrence_of ?occ2 (cut ?x))
                               (subactivity_occurrence ?occ1 ?occ)
                               (subactivity_occurrence ?occ2 ?occ)
                               (min_precedes ?occ1 ?occ2)))))
```

Queries for Process Routing

- Does there exist a process in which a subactivity A_1 always occurs before a subactivity A_2?

$$(\exists a)\ activity(a) \wedge ((\forall o, o_1, o_2)\ occurrence\_of(o, a)$$

$$\wedge occurrence\_of(o_1, A_1) \wedge occurrence\_of(o_2, A_2)$$

$$\wedge subactivity\_occurrence(o_1, o) \wedge subactivity\_occurrence\_of(o_2, o)$$

$$\supset min\_precedes(o_1, o_2, a)$$

Reasoning about Effects

1. Does every occurrence of a given (complex) activity change a given fluent?

$$(\forall o)\, occurrence\_of(o, A) \supset changes^*(o, F)$$

2. Is it possible to change a given fluent after performing a given activity?

$$(\forall o_1)\, occurrence_of(o_1, A) \supset (\exists o_2)\, precedes^*(o_1, o_2) \wedge changes^*(o_2, f)$$

3. Find (complex) activities that change the same fluents as a given activity.

$$(\exists a)(\forall o_1, o_2)\, occurrence\_of(o_1, A) \wedge occurrence\_of(o_2, a)$$

$$\supset (changes^*(o_1, f) \equiv changes^*(o_2, f))$$

Results

- We successfully answered a set of queries (including those above). However ...
- Automated reasoning is difficult!
- Other queries were not answered in the time limit (1800 seconds).

Approach

- Not all axioms are created equal
- Find minimal subsets of the ontology
- Use lemmas
- Complexity analysis

Axioms as Integrity Constraints

- Some axioms are used as integrity constraints on process descriptions, and such axioms are not used to prove theorems.

The subactivity relation is a discrete ordering, so every activity has an upwards successor in the ordering.

```
(forall (?a1 ?a2)
        (if     (subactivity ?a1 ?a2)
                (exists (?a3)
                        (and    (subactivity ?a1 ?a3)
                                (subactivity ?a3 ?a2)
                                (forall (?a4)
                                        (if     (and    (subactivity ?a1 ?a4)
                                                        (subactivity ?a4 ?a3)
                                                (or     (= ?a4 ?a1)
                                                        (= ?a4 ?a3)))))))))
```

Minimal Subsets of the Ontology

- Not all of the axioms of the ontology are used in proofs.
- Approach 1:
  - Extract the axioms that are used in the proofs of a set of queries.
- Approach 2:
  - Find axioms that are responsible for the combinatorial explosion by adding axioms from the ontology back to the minimal subsets.

Axioms that Cause Problems

Distinct occurrences of an activity correspond to distinct branches of an activity tree.

```
(forall (?a ?s1 ?occ1 ?occ2)
        (if     (and    (occurrence_of ?occ1 ?a)
                        (occurrence_of ?occ2 ?a)
                        (not (= ?occ1 ?occ2)))
                (exists (?s1 ?s2)
                        (and    (arboreal ?s1)
                                (arboreal ?s2)
                                (not (min_precedes ?s1 ?s2 ?a))
                                (not (min_precedes ?s2 ?s1 ?a))
                                (subactivity_occurrence ?s1 ?occ1)
                                (subactivity_occurrence ?s2 ?occ2)))))
```

Using Lemmas

- Some sentences, which are not axioms of the ontology, are used in proofs.
- These sentences are themselves difficult to prove.
- If we explicitly include these sentences together with the axioms, proofs become much shorter.
- Example: The *min_precedes* relation is transitive.

```
(forall (?s1 ?s2 ?s3 ?a)
   (if   (and   (min_precedes ?s1 ?s2 ?a)
                (min_precedes ?s2 ?s3 ?a))
         (min_precedes ?s1 ?s3 ?a)))
```

Complexity Analysis

We can prove that some queries are tractable for restricted classes of activities.

Theorem

Suppose the complex activity P has only finite activity trees. There exists an $O(n^2)$ algorithm to determine

$$T_{psl} \cup \Sigma_{pd}(P) \cup SPA \models (\forall o)\, root(o, P) \supset$$

$$(\exists o_1, o_2) occurrence\_of(o_1, A_1) \wedge min\_precedes(o, o_1, P)$$

$$\wedge occurrence(o_2, A_2) \wedge min\_precedes(o, o_2, P)$$

$$\wedge min\_precedes(o_1, o_2, P)$$

where n is the number of existentially quantified activity occurrence variables in $\Sigma_{pd}(P)$.

Summary

- A first-order process ontology can be used to create smart objects that can reason about the manufacturing processes in which the object participates.
- Focus on the development of techniques for efficient automated reasoning.