CSCI427/927 Service-Oriented Software Engineering

Service compliance management

Organizational compliance management

- Ensuring that the activities of an organization satisfy ***a set of rules***
  - Sources of these rules:
    - Legislation/Regulatory frameworks
      - General Data Protection Regulation (GDPR) in the European Union
      - Basel III framework for banking supervision
    - Business rules
      - Internal company policies for credit card processing
- The activities of an organization are represented as:
  - Process models/workflows (design-time)
  - Process instances (run-time)

Organizational compliance management (cont.)

- Compliance management has emerged as a major problem following major corporate governance scandals (e.g. Enron, WorldComm) and the resulting legislation (e.g. Sarbanes-Oxley Act)
- <u>Cost of compliance management</u> is very high
- Compliance software industry has become very large
- Most compliance software products are limited in functionality – primarily focusing on maintaining process transaction logs
  - Compliance software typically <u>records all activities and transactions</u> that occur within a system <u>to ensure a traceable record</u> for audit purposes

Compliance analyses

- **Compliance discovery:** What are our compliance obligations?
  - A healthcare organization **discovers its compliance obligations** under the **Health Insurance Portability and Accountability Act (HIPAA)**, which mandates strict privacy and security rules for handling patient data. They identify that all patient records must be encrypted and access restricted to authorized personnel.
- **Compliance modelling:** How do we effectively represent compliance requirements, service and process designs and enterprise structures that they impinge upon?
  - Compliance modelling **involves creating a process model** that visually and functionally integrates compliance rules into workflows, systems, or services. The model ensures that all steps follow necessary legal and business requirements.

Compliance analyses

- **Compliance checking:** Are we compliant? Where are the compliance violations? => involves **auditing or monitoring** processes to determine

- **Compliance resolution:** What do we do if we are non-compliant? How do we fix compliance violations? How might we identify the “best” way to modify process models deemed to be non-compliant?
  - After discovering non-compliance with PCI-DSS, the e-commerce company **upgrades its payment gateway** to ensure encryption for stored credit card data and conducts staff training on secure payment handling practices.

- **Compliance change management:** How do we manage changing compliance obligations? How do we maintain compliance in the dynamic business contexts
  - This involves updating compliance strategies and systems when regulations change

Compliance analyses

- **Compliance-by-design:** How do we support compliance-driven process design? => embedding regulatory requirements directly into the design of services, processes, or systems to ensure that compliance is an inherent feature, rather than something checked afterward.

- **Compliance monitoring:** Is every process execution compliant? How compliant are we?
  - This involves the continuous monitoring of processes to ensure that they comply with the required rules.

- These examples demonstrate how organizations manage different aspects of compliance across industries, from discovery to continuous monitoring.

Structural vs. Semantic non-conformance

- *Structural non-conformance*:
  - activities are overlooked or executed in the **wrong order**, or the wrong activities are executed

- Semantic non-conformance:
  - situations where the execution of a process might be structurally correct (the right activities are executed in the right order), but the **effects achieved do not conform to what is required by design**, potentially due to human errors.

Example

- Consider a clinical process that requires the administration of an anti-hypertensive medication.
- Correct execution of this task would require that a nurse should deliver the medication to the patient in question and depart only when the patient has ingested the medication.
- A **semantically non-conformant execution** might occur if the nurse delivers the medication to the patient, but does not stay around to confirm that the patient has actually taken it (and the patient happens to not take the medication).
- In a hospital with a process-aware information system, the nurse might then confirm to the process engine that this task has been completed, leading to a situation where **no structural non-conformance** would be detected.

Example (cont.)

- The fact that this process instance is semantically non-conformant can only be determined by checking ***the effects* of the process** to ensure that what is expected is actually obtained.

$\Rightarrow$ For example, a <u>blood pressure check</u> later in the day might reveal elevated readings, when the expected readings are lower.

$\Rightarrow$ *Semantic non-conformance is flagged in settings where <u>the observed effects deviate</u> from the expected ones.*

Example (cont.)

- Semantic non-conformance can be “fixed” by introducing *human-mediated activities* constructed on the fly
    - In contrast, machine-mediated functionality, such as a new web service, can often take too long to be able to correct errors in an executing process instance
- In our example: semantic non-conformance detected via the blood pressure check can be fixed by having the nurse correctly administer the medication as soon as possible.
- Once this is done, the clinical process instance involving this patient would be restored to a semantically conformant state.

Example (cont.)

- We can also address the problem of computing the best “fixes” of this kind, which we shall refer to as ***compensations***.
- Non-trivial: Common-sense compensation in our example - administer the **anti-hypertensive medication** as soon as the elevated blood pressure is detected
  - But: this might not be possible because of potential interactions between the anti-bypertensive medication and a more recently administered drug.
- We might thus need to **search** through the space of possible process re-designs to identify the “best”

Semantic Process Monitoring

Semantically annotated process models

Definition

A semantically annotated process model P is a process model in which each activity or event is associated with a set of ***effect scenarios***. Each effect scenario es is a **4-tuple** $(ID, S, Pre, Succ)$, where S is **a set of sentences in the background language**, ID is a unique ID for each effect scenario, Pre is a set of IDs of effect scenarios that can be valid ***predecessors*** in P of the current effect scenario, while $Succ$ is a set of IDs of effect scenarios that can be valid ***successors*** in P of the current effect scenario.

Example

The activity "Payment" is a key activity, after which the order status changes. The effect scenario for this activity might include the following set of sentences:

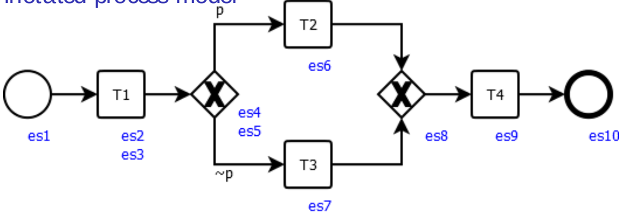
One S can include:

1. Paid(Order) (Indicates that the order has been paid)
2. ¬Cancelled(Order) (Indicates that the order has not been cancelled)
3. Processed(Payment) (Indicates that the payment has been processed)

These sentences use predicate logic to formally describe the system's state after the "Payment" activity.

Example

Semantically annotated process model



Annotation:

- $(es_1, S_1, \emptyset \{ es_2, es_3 \})$
- $(es_2, S_2, \{ es_1 \}, \{ es_4 \})$
- $(es_3, S_3, \{ es_1 \}, \{ es_5 \})$
- $(es_4, S_2, \{ es_2 \}, \{ es_6 \})$
- $(es_5, S_3, \{ es_3 \}, \{ es_7 \})$
- $(es_6, S_4, \{ es_4 \}, \{ es_8 \})$
- $(es_7, S_4, \{ es_5 \}, \{ es_8 \})$
- $(es_8, S_4, \{ es_6, es_7 \}, \{ es_9 \})$
- $(es_9, S_5, \{ es_8 \}, \{ es_{10} \})$
- $(es_{10}, S_5, \{ es_9 \}, \emptyset)$

Sentences:

- $S_1 = \emptyset$
- $S_2 = \{p, q\}$
- $S_3 = \{\neg p, q\}$
- $S_4 = \{p, q, r\}$
- $S_5 = \{p, \neg q, r\}$

Semantic Process Monitoring

Normative traces

Definition

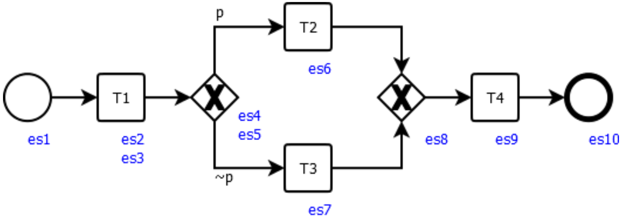
A normative trace *nt* is a sequence $(\tau_1, es_1, \tau_2, \ldots es_{n-1}, \tau_n, es_n)$, where

- $es_i \ldots, es_n$ are effect scenarios, and for each $es_i = (ID_i, S_i, Pre_i, Succ_i)$, $i \in [2..n]$, it is always the case that $ID_{i-1} \in Pre_i$ and $ID_i \in Succ_{i-1}$;
- $es_n = (ID_n, S_n, Pre_n, \emptyset)$ is the *final effect scenario*, normally associated with the end event of the process;
- $es_1 = (ID_1, S_1, \emptyset Succ_1)$ is the *initial effect scenario*, normally associated with the start event of the process;
- Each of $\tau_1, \ldots, \tau_n$ is either an event or an activity in the process.

We shall refer to the sequence $(\tau_1, \tau_2, \ldots, \tau_n)$ as the ***identity*** of the trace *nt*.

Example

Normative trace



Normative traces:

- $(START, es_1, T1, es_2, GATE1, es_4, T2, es_6, GATE2, es_8, T4, es_9, END, es_{10})$
- $(START, es_1, T1, es_3, GATE1, es_5, T3, es_7, GATE2, es_8, T4, es_9, END, es_{10})$

Semantic Process Monitoring

Definition

A semantic execution trace of a process P is a sequence $(\tau_1, o_1, \tau_2, o_2, \ldots, \tau_m, o_m)$ where each τ_i is either a task or an event, and **each o_i is a set of sentences in the background language** that we shall refer to as an *observation* that describes (possibly incompletely) the state of the process context after each task or event. We shall refer to the sequence $(\tau_1, \tau_2, \ldots, \tau_m)$ as the *identity* of the execution trace.

Key Differences

Aspect	Normative Trace	Semantic Execution Trace
Focus	What **should happen** (according to norms or rules)	What **happens** (with semantic details and effects)
Purpose	Describes the ideal or expected process sequence	Captures actual execution with semantic meaning
Determinism	Generally deterministic, following a predefined path	May handle non-determinism (e.g., different paths)
Relation to Process	Describes the normative model or regulation for the process	Describes real-time or simulated process execution

Semantic Process Monitoring

Semantic non-conformance and violation point

Definition

An execution trace $et = (\tau_1, o_1, \ldots, \tau_m, o_m)$ is said to be non-conformant with respect to a semantically annotated process P if and only if any of the following hold: (1) There exists an o_i in et such that for all normative traces $nt^{\prime} = (\tau^{\prime}_1, es_1, \ldots, \tau^{\prime}_i, es_i, \ldots)$ for which the identity of $(\tau_1, o_1, \ldots, \tau_i, o_i)$ is a prefix of its identity and $o_j \models es_j$ for each $j = 1, \ldots, i-1$, $o_i \not\models es_i$ (we shall refer to this as *weak semantic non-conformance*). (2) If we replace non-entailment with inconsistency in condition (1) above, i.e., $o_i \cup es_i \models \perp$, we obtain *strong semantic non-conformance*. In each case, we shall refer to τ_i as the violation point in the process.

To simplify of the presentation, from here, every *es* in the trace refers to *S* in the 4-tuple *(ID, S, Pre, Succ)* because *ID*, *Pre*, and *Succ* are meta-information used only to construct normative traces.

Semantic non-conformance and violation point

Weak/Strong Semantic Non-Conformance:

- **In simpler terms:** This means that the observed outcome $\mathbf{o_i}$ for activity T_i **is (not match)/(inconsistent with)** the expected outcome for the same activity in any normative trace. However, this non-conformance is **not considered a contradiction or impossible**, just a mismatch.
- In both cases, the activity T_i is the **violation point.**
- **Example:**
  - If a payment process is expected to result in "Order Paid" (in the normative trace), but the actual outcome is "Payment Failed" (in the execution trace)
  - If an order status is shown to be both "Shipped" and "Not Shipped" simultaneously, this would result in strong semantic non-conformance, as the outcomes contradict each other.

Semantic Compensation

Semantically compensated process instance

Definition

A process instance $et = (\tau_1, o_1, \ldots, \tau_m, o_m)$ will be referred to as a semantically compensated instance of a (semantically annotated) process P if there exist τ_i and τ_j in et, with $i < j$, such that τ_i is a violation point, and there exists a normative trace $nt = (\tau_1, es_1, \tau_2, \ldots es_{h-1}, \tau_h, es_h, \ldots, \tau_n, es_n)$ of P with an identity for which $(\tau_1, \ldots, \tau_{j-1})$ serves as a prefix, such that $o_k \models es_l$ for $k = j, \ldots, m$ and $l = h, \ldots, n$. As well, it must be the case that $o_m \models g$. We shall refer to τ_j as the *compensation point*<u>**. The compensation point must be a task and not an event.**</u>

We view process instances as semantic execution traces. We also assume that each process is associated with a *goal assertion g*.

Semantically compensated process instance

A later **compensation point**, T_j (with i<j), where actions are taken to recover from or correct the violation.

Imagine an online order process:

1. The process follows the normal steps (e.g., order creation, payment) up to a violation point where **payment fails**.
2. After detecting the failure, the system takes **corrective action** by retrying the payment or using an alternative payment method (this is the **compensation point**).
3. After the compensation point, the process continues normally (e.g., confirming payment, shipping the order), and the final goal (successful order completion) is achieved.

Semantic Compensation

Definition

Given a semantically compensated process instance $et = (\tau_1, o_1, \ldots, \tau_m, o_m)$ of P with a compensation point τ_j, ***a compensation is a process design*** $P^{\prime}$ for which the **completion of τ_{j-1} serves as the start event and $(\tau_j, o_j, \ldots, \tau_m, o_m)$ is a valid normative trace**. Every normative trace associated with $P^{\prime}$ must end in an effect scenario es such that $es \models g$, where g is the goal associated with the original process P.

Example

Violation and Compensation

Partial execution trace:

$$(START, o_1, T1, o_2, GATE1, o_4, T2, o_6)$$

where $o_6 = (\neg p, q, r)$.
A normative trace with the same prefix is

$$(START, es_1, T1, es_2, GATE1, es_4, T2, es_6, GATE2, es_8, T4, es_9, END, es_{10})$$

where the expected effect scenario according (es_6) should be $\{p, q, r\}$.
Intuitively, one possible compensated instance would be

$$(START, o_1, T1, o_2, GATE1, o_4, T2, o_6, T3, es_7, GATE2, es_8, T4, es_9, END, es_{10})$$

which is equivalent to *restart the precess at the XOR gateway* (there might be several other compensations).
How can we decide which compensated instance is better?

The computation of compensations

- One approach is to stop the execution of the current process and restart with a modified, re-designed version that includes reparations (alternatives to fulfill the original goal).

- Consider a reparation chain $O_1 \times O_2 \times \ldots \times O_m$
  - Each reparation/contrary-to-duty-obligation is an alternative means of achieving a goal
  - *“If we can’t put you on this flight, we’ll put you on the next one.”’*

  - Sometimes the reparation offers an alternative realization of an OR-parent goal of the current one. *“ If we can’t fly you to your holiday destination, we’ll get you there by train and give you 5 extra nights at the hotel”*

- Ultimately, goal realization (fulfilling the intended objective) and process proximity (how close the alternative process is to the original one) are key factors in designing effective reparations.

The computation of compensations

- One current approach considers settings where the process continues to execute even when semantic non-conformance is detected
- There is a trade-off to negotiate between the computation of *optimal* compensation, and the time the process is allowed to execute in a semantically non-conformant manner

- Imagine an online food delivery service where a customer orders a meal, but the restaurant cancels the order because the meal is no longer available.
    - It instantly suggests a similar meal from another restaurant with the same price
    - It offers a complimentary side or dessert, as well as a discount on the next order