A Strict-Observational Interface Theory for Analysing Service Orchestrations

Philip Mayer, Andreas Schroeder and Sebastian S. Bauer
LMU München, Germany
Motivation

- Domain: SOA Systems
  - (Self-Describing) Services with Protocols
  - Service Orchestrations

- Hard to design
  - Motivation: Provide formal support to developers wrt. *Self-Description* and *Orchestration*
Questions to be asked...

1. Does the service orchestration **conform** to the protocols?
2. Are the protocols of connected services **compatible**?
Answering Questions

• ...by using **Modal I/O-Transition Systems** as formalism
• ...and appropriate **Interface Theories**
Goal of this Work

• Existing interface theories for MIOs: „Strong“, „Weak“ and „May-Weak“ [TACAS‘10] based on (various notions of) modal refinement and output-compatibility

• Extensive case studies have shown: Hard to design, developers need to be assisted

• 2 sources of errors: protocol breaches and race conditions

• Our goal: A strict-observational interface theory (complementing other theories) for the early support of designers of interface specifications by only checking for protocol breaches.
Modal I/O-Transition Systems (MIOs)

(Larsen et al. 2007)

must-transition

may-transition

input action

output action

such that every **must**-transition is also a **may**-transition

\[ in = \{a\} \]
\[ out = \{b\} \]
\[ act = in \cup out \]
Modal Refinement: Idea

(Larsen, Thomsen 1988)

\[ S \leq m \ T \]

Orchestration  Protocol

• every **required (i.e. must)** transition in \( T \) is also **required** in \( S \)

• every **allowed (i.e. may)** transition in \( S \) is also **allowed** in \( T \)
Strict-Observational Refinement: Idea

\[ S \preceq_{so} T \]

Orchestration \quad Protocol

• every **required (i.e. must)** transition in \( T \) must also be **required** in \( S \), possibly preceded by **must**-transitions labelled with actions in
  \[ \text{act}_S \setminus \text{act}_T \]

• every **allowed (i.e. may)** path in \( S \), ending with an action in \( \text{act}_T \), possibly preceded by actions in \( \text{act}_S \setminus \text{act}_T \), must also be **allowed** by \( T \)

\( \text{in}_S \supseteq \text{in}_T \)
\( \text{out}_S \supseteq \text{out}_T \)
Example

Thesis Management

To Tutor
To Ex. Office
To Student

assessThesis!
update?
cancel?
complete?
abort!

Refinement of

≤ SO

update? complete? abort!
Example

Thesis Management

To Tutor
To Ex. Office

To Student

assessThesis!
complete?
update?
cancel?
update? complete? abort!

Refinement of

≤ so

assessThesis!
complete?
update?
cancel?
update? complete? abort!

update? complete? abort!
Example

Thesis Management

To Tutor

To Ex. Office

To Student

assessThesis!

update?

complete?

cancel?

abort!

update?

complete?

abort!

update?

complete?

abort!

Refinement of

\[ \leq \]

\[ so \]
Strict-Observational Refinement: Definition

Let $S$ and $T$ be MIOs such that $\text{in}_S \supseteq \text{in}_T$, $\text{out}_S \supseteq \text{out}_T$.

$S$ strict-observationally refines $T$, denoted by $S \leq_{so} T$, iff there exists a relation $R \subseteq \text{states}_S \times \text{states}_T$ containing $(\text{start}_S, \text{start}_T)$ such that for all $(s, t) \in R$, for all actions $a \in \text{act}_T$,

1. if $t \xrightarrow{a} T t'$, then there exists $s' \in \text{states}_S$ and $b_i \in (\text{act}_S \setminus \text{act}_T)$ such that $s \xrightarrow{b_1}_S \ldots \xrightarrow{b_n}_S \xrightarrow{a}_S s'$ and $(s', t') \in R$,

2. for all $b_j \in (\text{act}_S \setminus \text{act}_T)$, if $s \xrightarrow{b_1}_S \ldots \xrightarrow{b_m}_S \xrightarrow{a}_S s'$, then there exists $t' \in \text{states}_T$ such that $t \xrightarrow{a} T t'$ and $(s', t') \in R$. 
Strict-Observ. I/O Compatibility: Idea (1)

\[ S \sim_{so} T \]

We are interested in a **weak variant of output compatibility**: For every reachable state in \( S \otimes T \), directly after a synchronised communication between \( S \) and \( T \),

- if \( S \) **may** send out an action shared with \( T \), then \( T \) **must** be able to receive it, and
- conversely.
Strict-Observ. I/O Compatibility: Idea (2)

• Question: What do we get guaranteed by this notion of compatibility?

• If \( S \not\sim_{so} T \), then there is an output which the partner (or a refinement of the partner) is not able to receive in any execution of the system.
Example: Protocol to Protocol

Thesis Management

To Tutor

To Ex. Office

Student

compatible?

update?

complete?

abort!

update!

complete!

abort!

Compatible?

~\sim S_O

update?

complete?

abort!
Example: Orchestration to Protocol

Thesis Management

Student

To Tutor
To Ex. Office

assessThesis!

complete?
update?
cancel?

assessThesis!

compatible?

update!
complete!
abort?

update?
complete?
abort!
Example: Orchestration to Protocol

Thesis Management

Student

To Tutor
To Ex. Office

assessThesis!
cancel?
update?
complete?
abort!

compatible?

assessThesis!
cancel?
update?
complete?
abort!

update!
complete!
abort!

update?
complete?
abort!
Example: Orchestration to Protocol

Thesis Management

Student

To Tutor

To Ex. Office

assessThesis!

complete?

cancel?

update?

abort!

update!

complete!

abort?

compatibility

so

update?

complete?

abort!
Strict-Observational I/O Compatibility

Let $S$ and $T$ be (composable) MIOs.

$S$ and $T$ are called **strict-observationally I/O compatible**, written $S \sim_{so} T$, iff there exists a relation $R \subseteq states_S \times states_T$ containing $(start_S, start_T)$ such that for all $(s, t) \in R$,

1. for all $a \in (out_S \cap in_T)$, for all $b_i \in (act_S \setminus shared(S, T))$, if

   $$s \xrightarrow{b_1} \ldots \xrightarrow{b_n} \xrightarrow{a!} s',$$

   then there exists $t' \in states_T$ and $c_j \in (act_T \setminus shared(S, T))$ such that

   $$t \xrightarrow{c_1} \ldots \xrightarrow{c_m} \xrightarrow{a?} t'$$

   and $(s', t') \in R$,

2. and conversely, from $T$ to $S$. 
Preservation of Compatibility

• Theorem:

If $S \sim_{so} T$ and $S' \leq_{so} S$
and $\text{shared}(S, T) = \text{shared}(S', T)$, then $S' \sim_{so} T$. 
Tool Support: MIO Workbench

- Implements Strong, Weak, and Strict-Observational Refinement & Compatibility (see also [TACAS‘10])
- Example:
Overall Picture of the Approach

UML4SOA: UML profile for designing service protocols and orchestrations

- Developers design specifications on UML level
- Analysis stays on MIO level
- Automatic transformation & back-annotation in-between

=> „Hidden Formal Methods“

Early in the development process, strict-observational refinement and compatibility provides valuable feedback to the developer of UML4SOA diagrams.
Conclusion

• Motivation: Support of SOA developers
• Tool Support: MIO Workbench
• Strict-observational refinement & compatibility complements existing ref. & compatibility notions for MIOs

But still future work:
• Further support of developers on UML level
• Need for guaranteeing more properties like, e.g., input-compatibility, deadlock freeness
• Full automation and integration

See www.miowb.net for examples and implementation.
An interface theory is a tuple \((\mathcal{A}, \otimes, \leq, \sim)\) consisting of

- a domain \(\mathcal{A}\) of specifications
- a composition operator \(\otimes: \mathcal{A} \times \mathcal{A} \to \mathcal{A}\)
- a reflexive and transitive refinement relation \(\leq \subseteq \mathcal{A} \times \mathcal{A}\)
- a symmetric compatibility relation \(\sim \subseteq \mathcal{A} \times \mathcal{A}\)

satisfying

1. Compositional refinement:
   If \(S' \leq S\) and \(T' \leq T\), then \(S' \otimes T' \leq S \otimes T\).

2. Preservation of compatibility:
   If \(S \sim T\) and \(S' \leq S\) and \(T' \leq T\), then \(S' \sim T'\).
Further Results

- **Theorem: Compositionality of Refinement**

If $S' \leq_{so} S$ and $\text{shared}(S, T) = \text{shared}(S', T)$, then $S' \otimes_{so} T \leq_{so} S \otimes_{so} T$.

for a strict-observational composition operator $\otimes_{so}$

\[
\begin{align*}
    s & \xrightarrow{b_1} S \cdots \xrightarrow{b_n} S \xrightarrow{a} S' \quad a \in (\text{act}_S \setminus \text{shared}(S, T)) \\
    t & \xrightarrow{b_1} T \cdots \xrightarrow{b_n} T \\
    (s, t) & \xrightarrow{a} S \otimes_{so} T \quad (s', t')
\end{align*}
\]
Designing SOAs in UML

Orchestration (Activity)

Protocol (PSM)

Thesis Management