Contracts: Model-centric Assumption Promise System Specification

Manfred Broy
A System and its Operational Context
Operational Context (OPC)

User Interface

Physical and technical context

System under Consideration (SYS)

ASU \land PRO \Rightarrow IAS

Operational Context (ASU)
Assumptions for the requirements specification

Interaction Observations (IAS)
Properties for the observations in the requirements specification

System (PRO)
Properties promised in the system specification
Requirements Specification: Modeling SYS, OPC, OBS

• formulating system properties – system promises PRO
• formulating properties of the operational context – context assumptions ASU
• formulating properties of the interaction between the system and its operational context – interaction assertions IAS

\[
\text{ASU} \land \text{PRO} \Rightarrow \text{IAS}
\]

• This leads to assumption/promise specification formats
Discrete systems: the system modeling theory

Sets of typed channels

\[ I = \{x_1 : T_1, x_2 : T_2, \ldots \} \]
\[ O = \{y_1 : T'_1, y_2 : T'_2, \ldots \} \]

syntactic interface

\( (I \triangleright O) \)

data stream of type \( T \)

\[ \text{STREAM}[T] = \{\mathbb{N}\backslash\{0\} \rightarrow T^*\} \]

valuation of channel set \( C \)

\[ [C] = \{C \rightarrow \text{STREAM}[T]\} \]

interface behaviour for syn. interface \( (I \triangleright O) \)

\[ [I \triangleright O] = \{[I] \rightarrow \wp([O])\} \]
Example: System interface specification

A question answering component

QLC

\[
\begin{align*}
\text{in} & \quad x : Qst \\
\text{out} & \quad y : Asw \\
\forall q \in Qst: \quad q \# x = A[q] \# y
\end{align*}
\]

Every question gets answered!

Qst : set of questions
Asw : set of answers
A[q] : set of possible answers for question q

\[q \neq q' \Rightarrow A[q] \cap A[q'] = \emptyset\]
Verification: Proving properties about specified systems

From the interface assertions we can prove

• Safety properties

\[ a \in y \land y \in QAC(x) \Rightarrow \exists q \in Qst: q \in x \land a \in A[q] \]

• Liveness properties

\[ q \in x \land y \in QAC(x) \Rightarrow \exists a \in Asw: a \in y \land a \in A[q] \]
Example: Context interface specification

A context interface specification

\[
\begin{align*}
\text{CIS} \\
\text{in} & \quad y : \text{Asw} \\
\text{out} & \quad x : \text{Qst} \\
\forall \ t \in \text{Time}: & \quad \text{Qst}(x \downarrow t) + 1 \leq \text{Asw}(y \downarrow t)
\end{align*}
\]

Never ask a further question before your recent one is answered.
Example: QAC with Timing Restrictions

\[
\forall t \in \mathbb{IN}: \forall q \in Qst:
\]
\[
A[q](y \downarrow t + \text{delay})
\]
\[
\leq q'(x \downarrow t)
\]
\[
\leq A[q](y \downarrow t + \text{delay} + \text{deadline})
\]
Universal Properties of System (Interfaces)
System interface behaviour - causality

\((I \rightharpoonup O)\) syntactic interface with set of input channels \(I\) and of output channels \(O\)

\(F \in [I \rightharpoonup O]\) semantic interface for \((I \rightharpoonup O)\) with timing property addressing strong causality (let \(x, z \in [I], y \in [O], t \in \mathbb{N}\):

\[
x_{\downarrow t} = z_{\downarrow t} \Rightarrow \{y_{\downarrow t+1}: y \in F(x)\} = \{y_{\downarrow t+1}: y \in F(z)\}
\]

\(x_{\downarrow t}\) prefix of history \(x\) of length \(t\)

Component interface

\(I\) \hspace{1cm} \(O\)
Verification: adding/exploiting causality

From the interface assertions we can derive properties!

Specification:

\[ y \in QAC(x) \Rightarrow (\forall q \in Qst: \; q\#x = A[q]\#y) \]

Strong causality: \( \forall t \in \text{Time}: \)

\[ x\downarrow t = z\downarrow t \Rightarrow \{y\downarrow t+1: y \in QAC(x)\} = \{y\downarrow t+1: y \in QAC(z)\} \]

From which by choosing \( z \) such that

\[ #(z\uparrow t) = 0 \]

we can deduce (note then \( q\#x\downarrow t = q\#z \))

\[ y \in QAC(x) \Rightarrow \forall q \in Qst: A[q]\#(y\downarrow t+1) \leq q\#(x\downarrow t) \]

No answers before questions!
Causal deterministic behaviors

• A total function $f: [I] \rightarrow [O]$ is called causal (and strongly causal, respectively) if behaviour
  $$F \in [I \triangleright O] \text{ with } F(x) = \{f(x)\}$$
  is causal (or strongly causal, respectively) for all $x \in [I]$

• A nondeterministic behaviour $F$ defines the set $[F]$ of total deterministic behaviours.
An interface behaviour $F$ is called *(strongly) realizable* if there exists a (strongly) causal “deterministic” function $f: [I] \rightarrow [O]$ such that
\[ \forall x \in [I]: f(x) \in F(x) \]
f is called (strong) realization of $F$.

**Theorem**

An interface behaviour $F$ is *(strongly) realizable* if there exists a *(Moore) Mealy machine* that calculates $F$. 
Example: Non-realizable Behaviour

Consider the behaviour $F \in [I \mapsto O]:$

$$F(x) = \{y \in [O] : x \neq y\}$$

$F$ is strongly causal but not realizable.

**Proof:** Strong causality is obvious.

If $F$ were realizable $f \in [F]$ exists with

$$\forall x \in [I] : f(x) \in F(x)$$

Since $f$ is strongly causal there exists a fixpoint $z$ with $z = f(z)$.

By $f \in [F]$ we get by $y = f(x)$ the proposition $y \in F(x)$ and by the specification $x \neq y$ and thus for the fixpoint $z$ the conclusion $z \in F(z)$ which yields $z \neq z$ and thus a contradiction.
Accordingly, for an interface assertion \( \text{spc}(x, y) \) the following healthiness conditions are required:

Existential satisfiability: \( \forall x: \exists y: \text{spc}(x, y) \)

Strong causality
in input \( x \):
\[
\forall x, x': \forall t: x \downarrow t = x' \downarrow t \Rightarrow \forall y: \text{spc}(x, y \downarrow t+1) = \text{spc}(x', y \downarrow t+1)
\]

Realizability:
\[
\exists f \in \text{If}_{\text{sc}[I \triangleright O]}: \forall x: \text{spc}(x, f(x))
\]

Full realizability:
\[
\forall x, y: \text{spc}(x, y) \Rightarrow \exists f \in \text{If}_{\text{sc}[I \triangleright O]}:
\quad y = f(x) \land \forall x': \text{spc}(x', f(x'))
\]
We model
• context behavior by (weakly) causal behaviors assuming that contexts may react instantaneously – reaction within one time interval
• system by strong causal behavior – reaction requires at least one step in time

Consequence:
Unique fixpoints for realizations of the context and the system specs
Healthiness Conditions for System Context Specifications

Accordingly, for an interface assertion \( \text{asu}(x, y) \) of the context the following healthiness conditions are required:

Existential satisfiability: \( \forall y: \exists x: \text{asu}(x, y) \)

Weak causality

in input \( x \):

\( \forall y, y': \forall t: y \downarrow t = y' \downarrow t \Rightarrow \forall y: \text{asu}(x \downarrow t, y) = \text{asu}(x \downarrow t, y') \)

Realizability:

\( \exists g \in \text{If}_c[O \triangleright I]: \forall y: \text{asu}(g(y), y) \)

Full realizability:

\( \forall x, y: \text{asu}(x, y) \Rightarrow \exists g \in \text{If}_c[O \triangleright I]: x = g(y) \wedge \forall y': \text{asu}(g(y'), y') \)

Context may react immediately in the current time interval
Modularity: Rules of compositions for interface specs

We do not need assumptions to achieve modularity!

\[ F_1 \otimes F_2 \]

\[ \begin{array}{l}
\text{in} \ x_1, z_{12}: T \\
\text{out} \ y_1, z_{12}: T \\
S_1
\end{array} \]

\[ \begin{array}{l}
\text{in} \ x_2, z_{12}: T \\
\text{out} \ y_2, z_{21}: T \\
S_2
\end{array} \]

F1 \otimes F2

\[ \begin{array}{l}
\text{in} \ x_1, x_2: T \\
\text{out} \ y_1, y_2: T \\
\exists z_{12}, z_{21}: S_1 \wedge S_2
\end{array} \]

S_1 \wedge S_2 is called the interaction assertion
Qsts and answers

System

\[
\forall t \in \text{Time}: \quad \text{Qst}(x \downarrow t) + 1 \leq \text{Asw}(y \downarrow t)
\]

Context

\[
\forall q \in \text{Qst}: \quad q \# x = A[q] \# y
\]

Interaction assertion:

\[
\forall t \in \text{Time}: \quad \text{Qst}(x \downarrow t) + 1 \leq \text{Asw}(y \downarrow t)
\]
\[
\land \forall q \in \text{Qst}: \quad q \# x = A[q] \# y
\]
Assumption/Promise (A/P) Specifications

Assumption: \( \text{asu}(x, y) \)
The properties that we assume about the interface behavior of a context

Promise: \( \text{pro}(x, y) \)
The properties that are guaranteed about the interface behavior of the system

Resulting system spec: \( \text{asu}(x, y) \Rightarrow \text{pro}(x, y) \)
Example: Assumption promise system interface specification

A contract for a question answering component

APQAC

<table>
<thead>
<tr>
<th>in</th>
<th>x: Qst</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>y: Asw</td>
</tr>
</tbody>
</table>

assumption

\[ \forall t \in \text{Time}: \ Qst#(x\downarrow t)+1 \leq \text{Asw#}(y\downarrow t) \]

promise

\[ \forall q \in \text{Qst}: q#x = A[q]#y \]

Every question gets answered - as long as the next question is answered only after all questions have been answered!
What is a good (a “healthy”) assumption?
Why Assumptions are Constraint by Output Histories

• In the general case, assumptions refer to output of the system.
  The reason is that if a system is **nondeterministic** and the question which input $x$ fulfils the assumption may depend on the actual output $y$ produced so far.

• Example: our QAS

  $\text{asu}(x, y) \equiv \forall t \in \text{Time}: \ Qst#(x\downarrow t)+1 \leq \text{Asw#}(y\downarrow t)$

  $\text{pro}(x, y) \equiv \forall q \in \text{Qst}: \ q#x = A[q]\#y$

  We obtain the specification in terms of an interface assertion

  $\text{con}(x, y) \equiv [\text{asu}(x, y) \Rightarrow \text{pro}(x, y)]$

  The assumption is fulfilled

  ◦ if a question is never sent as input to the system
  ◦ before the answer to the previously question has been returned by the system as output.
What makes an Interface Assertion a Healthy Assumption

• **Assumptions** should only constrain properties of the context.
  ◊ In the case of simple assumptions that only refer to the input histories \( x \in [I] \) for systems with systematic interface \( (O \triangleright I) \) this is obvious.

• However, what does it mean that **asu** only constraints the input histories for general assumptions.

\[
asu : [I] \times [O] \rightarrow IB
\]
Healthiness Conditions for Context Specifications

Accordingly, for an interface assertion \( \text{asu}(x, y) \) the following healthiness conditions are required:

Existential satisfiability: \( \forall y: \exists x: \text{asu}(x, y) \)

Causality in input \( y \): \( \forall y, y': \forall t : y \downarrow t = y' \downarrow t \Rightarrow \forall x: \text{asu}(x \downarrow t, y) = \text{asu}(x' \downarrow t, y) \)

Realizability: \( \exists g \in \text{If}_{c}[O \triangleright I]: \forall x: \text{asu}(g(y), y) \)

Full realizability: \( \forall x, y: \text{asu}(x, y) \Rightarrow \exists g \in \text{If}_{c}[O \triangleright I]: x = g(y) \land \forall y': \text{asu}(g(y'), y') \)
Example: Implicative Assertions

Consider a system with one input channel $x$ and one output channel $y$, both carrying natural numbers as messages. Let $n$ be a given natural number.

A specification in implicative form:

$$\text{con}(x, y) \equiv [n\#y = 0 \Rightarrow n\#x = 0]$$

Clearly, there does not exist a context that can guarantee the premise $n\#y = 0$, since the output is exclusively determined by the system.

Is $n\#y = 0$ a healthy assumption about the context?
**Example: Implicative Assertions**

The A/P–specification

- **assume:** \( n\#y = 0 \)
- **promise:** \( n\#x = 0 \)

is not healthy, since

- the assumption does not constrain the input histories but the output.
- The promise \( n\#y = 0 \) is not healthy as an assumption, since it does not express properties of input stream \( x \) but only of output stream \( y \).

The assertion \( n\#y = 0 \) is not causal in history \( y \), since causality in \( y \) would require for all \( t \in \text{IN} \)

\[
y \downarrow t = y' \downarrow t \Rightarrow \forall x: (n\#y = 0) \equiv (n\#y' = 0)
\]

which does not hold.

Assertion \( n\#y = 0 \) is therefore not a *healthy* assumption, since it is not causal in \( y \) and thus not realizable by any context.
Example: Implicative Assertions

In the assertion (which is equivalent to assertion $\text{con}(x, y)$ by contraposition)

$$\text{con}(x, y) \equiv [n#x > 0 \Rightarrow n#y > 0]$$

the assertion $n#x > 0$ is causal in history $y$ since the formula

$$y\downarrow t = y'\downarrow t \Rightarrow \forall x: (n#x\downarrow t > 0) \equiv (n#x\downarrow t > 0)$$

holds. It is furthermore trivially realizable.

This interface assertion may therefore be rewritten in the A/P-format of a contract

**assume:** $n#x > 0$

**promise:** $n#y > 0$

with a healthy assumption.

**Conclusion:** Not every assertion is a healthy assumption.
From Interaction Assertions to Assumptions and Promises
Given IAS, can we derive ASU and PRO?

Operational Context (ASU)
Assumptions for the requirements specification

Interaction Assertion (IAS)
Properties for the observations in the requirements specification

System (PRO)
Properties captured in the system specification

ASU \land PRO \Rightarrow IAS
Throughout the presentation we use the following notation:

Given a predicate

\[ p: \mathbb{C} \rightarrow \mathbb{IB} \]

we extend for every time \( t \in \mathbb{IN} \) the predicate \( p \) also to finite histories \( x \) of length \( t \):

\[ p(x) \equiv \exists x' \in : x = x' \downarrow t \land p(x') \]
Let the *interaction assertion*
\[
i_{\text{as}}: \mathbb{I} \times \mathbb{O} \rightarrow \mathbb{I}B
\]
be given.
\[
i_{\text{as}}(x, y)
\]
is an assertion characterizing the interaction between the system \(S\) and its context \(E\) in terms of the histories \(x\) and \(y\).
∀ t ∈ Time: Qst#(x↓t) + 1 ≤ Asw#(y↓t)
∧ ∀ q ∈ Qst: q#x = A[q]#y
Can we derive from the interaction assertion:

\[ \forall t \in \text{Time}: \ Qst#(x \downarrow t) + 1 \leq Asw#(y \downarrow t) \]
\[ \land \ \forall q \in Qst: q#x = A[q]#y \]

the contract in terms of assumptions and promises for
From Interaction Assertions to Contracts

Given an interaction assertion $\text{ias}(x, y)$ we derive an A/P-specification for system $S$ with the weakest assumption by the following steps:

(1) Separate $\text{ias}$ into a safety and a liveness part
(2) Separate the safety part of $\text{ias}$ canonically into an assumption and a promise for system $S$
(3) Separate the liveness part of $\text{ias}$ into an assumption and a promise for system $S$
(4) Construct a contact being the A/P-specification of $S$ from the liveness and safety parts of the assumption and the promise.
From Interaction Assertions to Contracts: Safety

Deriving \( \text{asu}(x, y) \) and \( \text{pro}(x, y) \) from \( \text{ias} \) such that:

\[
\text{asu}(x, y) \land \text{pro}(x, y): \quad \text{ias}(x, y)
\]

\[
\neg \text{asu}(x, y): \quad \exists \, t \in \text{IN}: \, \text{ias}(x \downarrow t, y \downarrow t+1) \land \neg \text{ias}(x \downarrow t+1, y \downarrow t+1)
\]

\[
\neg \text{pro}(x, y): \quad \exists \, t \in \text{IN}: \, \text{ias}(x \downarrow t, y \downarrow t) \land \neg \text{ias}(x \downarrow t, y \downarrow t+1)
\]
From Interaction Assertions to Contracts: Safety

Derive promise \( \text{pro} \) and a assumption \( \text{asu} \) from property \( \text{ias} \)

\[
\text{asu}(x, y) \equiv [\text{ias}(x, y \downarrow 0) \land (\forall t: \text{ias}(x \downarrow t, y \downarrow t+1) \Rightarrow \text{ias}(x \downarrow t+1, y \downarrow t+1))]
\]

\[
\text{pro}(x, y) \equiv (\forall t: \text{ias}(x \downarrow t, y \downarrow t) \Rightarrow \text{ias}(x \downarrow t, y \downarrow t+1))
\]

To eliminate partiality according to the input restriction in assertion \( \text{ias}(x, y) \) derive from interaction assertion \( \text{ias}(x, y) \) the weaker interface assertion \( \text{con}(x, y) \) specified by contract

\[
\text{con}(x, y) \equiv [\text{asu}(x, y) \Rightarrow \text{pro}(x, y)]
\]

An easy proof shows that \( \text{con}(x, y) \) is strongly causal.
Note that according to our initial assumption interaction assertion \( \text{ias}(x, y) \) includes only safety properties.

**Theorem:**
With the definitions as given above we obtain under the condition that assertion \( \text{ias}(x, y) \) is a pure safety property

\[
(\text{asu}(x, y) \land \text{pro}(x, y)) \Leftrightarrow \text{ias}(x, y)
\]
If $\text{ias}(x, y)$ includes nontrivial liveness conditions the separation into assumptions and promises of $\text{ias}(x, y)$ is less canonical than for safety, in general.

- Some liveness conditions definitely formulate properties specifically about input histories $x$ or histories $y$ about output.
- There are liveness conditions that can not be canonically separated into assumptions and promises.
- **Example:** the assertion
  \[ \{1\}#x + \{0\}#y = \infty \]
  can either be fulfilled by assuming an infinite number of copies of $1$ in input history $x$ or by promising an infinite number of copies of $0$ in output history $y$. 
From Interaction Assertions to Contracts: Liveness

Given an interaction assertion
\[ \text{ias}(x, y) \]
that is a pure liveness condition we define an assumption \( \text{asu}_{\text{ias}} \) as follows
\[ \text{asu}_{\text{ias}}(x) \equiv \exists y: \text{ias}(x, y) \]
and a promise \( \text{pro}_{\text{ias}} \) by the equation
\[ \text{pro}_{\text{ias}}(x, y) \equiv \text{ias}(x, y) \]
Those parts of the liveness property \( \text{ias} \) that can either be fulfilled by the context or by the system under consideration are made part of the promise.

\[ \Diamond \text{ This way we get the weakest assumption and the strongest promise for liveness properties of } \text{ias}. \]
We derive from the interaction assertion:

\[ \forall t \in \text{Time}: \ Qst \#(x \downarrow t) + 1 \leq Asw \#(y \downarrow t) \]
\[ \land \ \forall q \in Qst: q \# x = A[q] \# y \]

the specs for

**QAC**

\[ x : Qst \]

\[ y : Asw \]

**CIS**

\[ x : Qsts \]

\[ y : Asws \]
Deriving specs from interaction assertion

We derive from the interaction assertion:

\[
\forall t \in \text{Time}: \ Qst#(x \downarrow t) + 1 \leq Asw#(y \downarrow t) \\
\land \forall q \in Qst: \ q#x = A[q]#y
\]

the specs for the system interface and the context:
Safety property in \( x \) and \( y \)

\[
\forall t \in \text{Time}: \ Qst#(x \downarrow t) + 1 \leq Asw#(y \downarrow t)
\]

is clearly an assumption.
The property

\[
\forall q \in Qst: \ q#x = A[q]#y
\]

is composed of a system safety property (by causality)

\[
\forall t \in \text{Time}: \ \forall q \in Qst: \ q#x \downarrow t \geq A[q]#y \downarrow t + 1
\]

that is clearly a promise and liveness property

\[
\forall q \in Qst: \ q#x \leq A[q]#y
\]

that is turned into a promise.
Service Layers and Service Stacks
Assumed and Promised Services
Service Layers

A service layer is a service with syntactic interface \((I ∪ O' \downarrow I' ∪ O)\) structured into an promised ("exported") service \((I \downarrow O)\) and assumed ("imported") service \((I' \downarrow O')\).

We assume \(I \cap O' = \emptyset\) and \(O \cap I' = \emptyset\).

A service layer is a service with the interface behavior

\[ L \in [(I ∪ O') \downarrow (I' ∪ O)] \]

where both input and output actions are disjoint sets.
Service Layers

We denote the syntactic interface of a service layer by

$$(I \triangleright O/O' \triangleright I')$$ syntactic service layer interface

The service layer is a service

$L \in [I \cup O' \triangleright O \cup I']$
Specifying Service Layers

To specify a service layer we specify two services

The **promised service** \( F \in [I \rightarrow O] \)

The **assumed service** \( R \in [I' \rightarrow O'] \)

A layer provides the service \( F \) under the condition that it gets the service \( R \) from “below”.

Note that

◊ \( R \) does not specify the service of the layer as promised by \( L \) but the assumed service.

Given \( F \) and \( R \) we denote the layer \( L \) that offers service \( F \) provided service \( R \) is offered as an auxiliary service by

\[ F \parallel R \]
Specifying Service Layers

Layer

\[ L = F // R \]

is specified as follows (for \( x \in [I \cup O'] \))

\[ L(x) = \{ y \in [I' \cup O]: x|O' \in R(y|I') \Rightarrow y|O \in F(x|I) \} \]

This expresses that

- if the service assumed from "below" is correct as required and specified by \( R \) than the offered service is as promised by \( F \).
- Note that this specification is written in the pattern of an assumption/promise specification.
Composing Layers with Services

Given a service $R'$ as requested from below

\[ R' \in [I'\triangleright O'] \]

imported service

and a service layer

\[ L \in [I\triangleright O/O'/\triangleright I'] \]

where

\[ L = F//R \]

with given $F \in [I\triangleright O]$ and $R \in [I'/\triangleright O']$

We get the composition of layer $L$ with service $R'$

\[ L \otimes R' \in [I\triangleright O] \]

composition of layer $L$ with service $R'$
Service Refinement

Given services

\[ F, F' \in [I \rightarrow O] \]

\( F' \) is called a refinement of \( F \) iff

\[ \forall x \in [I]: F'(x) \subseteq F(x) \]

then we write

\[ F \rightarrow F' \]
Composing Layers with Services

\( L \otimes R' \in [I\rightarrow O] \) is a refinement of the provided service \( F \in [I\rightarrow O] \)

\[ F \rightarrow L \otimes R' \]

provided service \( R' \) is a refinement of the requested service \( R \)

\[ R \rightarrow R' \]

Given layer \( L = F//R \) we get the following proof rule for layered architectures

\[
\begin{align*}
L &= F//R \land R \rightarrow R' \\
\Rightarrow \\
F &\rightarrow L \otimes R'
\end{align*}
\]
Composing Layers

Given two layers
\[ L \in [I \Rightarrow O/O' \Rightarrow I'] , \quad L' \in [I' \Rightarrow O'/O'' \Rightarrow I''] \]
where we assume that \( I \cap I'' = \emptyset \) and \( O \cap O'' = \emptyset \);
we define the layer composition
\[ L \otimes L' \in [I \Rightarrow O/O'' \Rightarrow I'] \]
yielding a layer in \([I \Rightarrow O/O'' \Rightarrow I']\).

Assume service \( L \) and \( L' \) are described as follows
\[
L = F//R \\
L' = F'//R'
\]

We call these two layers fitting if \( R \rightarrow F' \)
then we conclude \( F//R' \rightarrow L \otimes L' \)

proof rule for layered architectures
\[
L = F//R \land L' = F'//R' \land R \rightarrow F' \quad \Rightarrow \\
F//R' \rightarrow L \otimes L'
\]
Assumption/Promise Contracts at several levels

- Assumptions about input
  - What are healthy assumptions?

- Assumptions about the behavior of the operational context
  - What are generic properties of the operational context (what is the model of context behavior)?

- Decomposing interactions into assumptions and system properties

- Assumptions in architectures
  - Show that validity of assumptions of the system guarantee all assumptions of components of the system the system!

- Assumptions about required services
  - Specifying and composing service layers!