Multiple viewpoint contracts in SPEEDS

SPEEDS Project Team
Airbus, Bosch, Carmeq, EADS, IAI, Knorr-Bremse, MagnaSteyr, Saab
Esterel Technologies, Extessy, Geensys, Telelogic
INRIA, OFFIS, Parades, Verimag

with contributions from Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud (INRIA Rennes), Alberto Ferrari, Leonardo Mangeruca, Christos Sofronis, Alberto Sangiovanni-Vincentelli (PARADES), and with input from Tom Henzinger (EPFL)

Presented by Roberto Passerone
PARADES / The University of Trento
Component-based design

- System partitioned into separate components
  - Reason individually about components
  - Study their interaction

- Separation between component specification and interaction making its way into mainstream methodologies
  - Orthogonalization of concerns reduces design complexity
  - Compositional techniques and composability the basis for an efficient and yet optimal design flow
  - Semantics of model based on parallel (or sequential) composition
Heterogeneous specification

Homogeneous components

Heterogeneous components

Viewpoint-based separation of concerns
Viewpoints

function

reliability

energy

timing

QoS
Component and viewpoint composition

Component-centric

Viewpoint composition

function

timing

reliability

energy

QoS

Component specification

Component specification

SYSTEM

Roberto Passerone, CPS week, St. Louis, MO
Component and viewpoint composition

- function
- timing
- reliability
- energy
- QoS

Viewpoint-centric

SYSTEM
Arbitrary nesting of operations

What are the operators?

What is the relation between the flows?
Establishing responsibilities

- Highly varied and fragmented design space
  - Components
  - Viewpoints
  - Interactions

- Must establish the responsibilities between the different actors precisely
  - Suppliers
  - System integrators
  - Users and environment

- Rely on early horizontal and vertical partitioning for speculative design
  - Concurrent development
  - Shorter design cycles, higher confidence
Address the above issues by

- providing a practical syntax to express component behavior and non-functional properties
- exploring the relations between contract-based, component-centric and viewpoint-centric composition operators
- providing language extensions through a layer-based approach
- developing innovative design and analysis techniques

Large integration project

- Academia, tool providers and users
- Will focus on the fundamental operators
- Will borrow terminology and syntax from familiar domains
Establishing responsibilities

- Express guarantees and assumptions of a component relative to its environment *(contract-based)*
  - Assumptions help simplify design by restricting the operating conditions
  - Guarantees used as hypothesis to help discharge system obligations

- Work on Assume/Guarantee reasoning [Hoare, Gries, Lamport, Meyer, Back & von Wright] and Interfaces [deAlfaro-Henzinger 01] have addressed some central issues
  - Substituability and the covariant/contravariant nature of refinement – assumptions must get weaker and guarantees stronger
  - How to compose contracts for different components
  - How to represent a contract in a computationally effective way, *e.g.*, *interface automata* by de Alfaro and Henzinger

- However, *refining a set of contracts* attached to the same component has not, to our knowledge, been considered so far
  - This is needed in order to address multiple viewpoints
Candidate framework: interface automata
[deAlfaro-Henzinger 01]

- I/O profile shown at the interface
- The automaton specifies who performs what in which state
- Assumption:
  - env. should never submit \textit{fail}
  - when in state 1, env. can submit \textit{ok}
- Guarantee: system responds as indicated
Candidate framework: interface automata
[deAlfaro-Henzinger 01]

The server

- I/O profile shown at the interface
- The automaton specifies who performs what in which state
- Assumption: environment offers what is allowed depending on the state
- Guarantee: system responds as indicated
Candidate framework: interface automata
[deAlfaro-Henzinger 01]

The result tells how the network should behave in order for the protocol to deliver service as expected by the client.
Adding viewpoints for the network

- The **client** wants to see again the same story: `msg!ok?msg!ok?...

- We have the same protocol component (the **server**), for reuse and possibly patch whenever needed

- In addition, we have some knowledge about the network

  - **functional**: when receiving a request `sent` from the server, it can answer either `ack` (in case transmission succeeds) or `nack` (otherwise)

  - **reliability**: the network will not stop failing to transmit until a `reset` action is performed by the supervisor
The corresponding architecture: multiple viewpoint

protocol

network-functional
network-reliability

client

Cannot be dealt with using the above construction
Modal Automata

[Kim Larsen 1989, Kim Larsen & al. 2007]

- Modal Automata are automata with \textit{may} and \textit{must} transitions
- Proposed by Kim Larsen in 1989 to study refinement specification and further compared to Interface Automata in 2007 in the context of product lines

1. $S :: \forall q: \text{must}(q) \subseteq \text{may}(q)$
2. $C \models S :: \forall q:
   \Rightarrow \text{must}(q) \subseteq C(q) \subseteq \text{may}(q)$
3. $S \preceq S' :: \forall q \sim q':
   \Rightarrow \text{must}(q) \supseteq \text{must}'(q')$
   \Rightarrow \text{may}(q) \subseteq \text{may}'(q')$
Modal Automata

- Explicit assumptions
  - Nominal behavior in black
  - Degraded behavior in red
  - Input enabled

- Exploit modalities
  - Keep colors as before, useful for interpreting nominal/degraded modes
  - Use modal automata with \textit{may/must} transitions
    - Inputs in nominal behavior are \textit{must} (input enabled)
    - Outputs in nominal behavior are \textit{may} (best effort)
    - Degraded mode is \textit{may}
Supporting substituability for multiple viewpoint contracts

- $S \land S'$
  - $\Rightarrow$ must$(q)$ $\cup$ must$'(q')$
  - $\Rightarrow$ may$(q)$ $\cap$ may$'(q')$

- $C \models S$ and $C \models S'$ iff
  $C \models S \land S'$
Supporting substituability for multiple components

- $S \otimes S'$
  - $\text{must}(q) \cap \text{must}'(q')$
  - $\text{may}(q) \cap \text{may}'(q')$
- $C \models S$ and $C' \models S'$ implies $C \times C' \models S \otimes S'$
- Collapse traps
Supporting substituability for multiple components

- $S \otimes S'$
  - $\text{must}(q) \cap \text{must}'(q')$
  - $\text{may}(q) \cap \text{may}'(q')$

- $C \models S$ and $C' \models S'$ implies $C \times C' \models S \otimes S'$

- Collapse traps
Multiple viewpoint contract for a system architecture

- protocol
- client

\[
\text{network-functional} \land \text{network-reliability}
\]
Relating the compositions

- The designer may want to
  - consider all viewpoints for each component
  - implement each component
  - compose the implementations

- Or, alternatively, consider viewpoints incrementally:
  - consider all viewpoints for each component except reliability and QoS
  - implement each component
  - compose the implementations
  - Revisit the design for reliability and QoS, possibly with a different, coarser grain, architecture
A fundamental theorem

\[
S \otimes (S' \land S'') \leq (S \otimes S') \land (S \otimes S'')
\]

Component centric \hspace{2cm} Viewpoint centric

Viewpoint Centric design leaves more room for implementations than Component Centric design
Acknowledgments

- End-users
  - Airbus, Bosch, Carmeq, EADS, IAI, Knorr-Bremse, MagnaSteyr, Saab

- Vendors
  - Esterel Technologies, Extessy, Geensys, Telelogic

- Research Organizations
  - INRIA, OFFIS, Parades, Verimag
Address the above issues by

- providing a practical syntax to express component behavior and non-functional properties
- exploring the relations between contract-based, component-centric and viewpoint-centric composition operators
- providing language extensions through a layer-based approach
- developing innovative design and analysis techniques

Large integration project

- Academia, tool providers and users
- Developing integration platform – the SPEEDS bus
- Understanding the relation between analysis and process