Strong and Weak Contract Formalism for Third-Party Component Reuse

Irfan Sljivo, Barbara Gallina, Jan Carlson, Hans Hansson

{Irfan.sljivo, barbara.gallina, jan.carlson, hans.hansson}@mdh.se
Safety and Component Reuse

● “3Cs Model” – Concept, Content, Context
● Safety is a system property

● Safety standards and reuse:
  ● DO-178B(C) – Reusable Software Component
  ● ISO 26262 – Safety Element out of Context
  ● Off-the-shelf items

If we include too much information about specific context or a system, we are reducing reusability of the component.
Component contracts

- Contract as a set of pre and post conditions between a client and a method with additional class invariants [Meyer92]

| ClassA | aMethod() | Client |
Component contracts

- Contract as a set of pre and post conditions between a client and a method with additional class invariants [Meyer92]

- Assumption/guarantee contracts $C = (A, G)$ [Benveniste07]
  - Assumptions and guarantees defined in terms of behaviours
Component contracts

- Contract as a set of pre and post conditions between a client and a method with additional class invariants [Meyer92]

- Assumption/guarantee contracts \( C=(A, G) \) [Benveniste07]
  - Assumptions and guarantees defined in terms of behaviours

- Extended A/G contracts \( C=(A, B; G, H) \) [Benveniste11]
  - Weak and strong assumptions and guarantees
  - Motivation for separation methodological
Contributions and Related Work

● Contributions
  ● Further development of the strong and weak contracts by adaptation of contract semantics
  ● Usage demonstration on a wheel braking system

● Related work
  ● Mostly theoretical works ([Benveniste et al], [Bauer et al])
  ● Focus on OEM-supplier relationship
Approach

Rich Component Concept

- Interface description
- Implementation
- Static analysis
- Safety Argumentation
- Contracts
- Test cases

Backning arguments
Trustworthiness arguments

Safety
Timing
Functional

...
Fine-grained Contracts

- The contract format:
  \[<A, G, \{<B_1, H_1>, \ldots, <B_n, H_n>\}>\]

- Strong assumptions must hold, strong guarantees always offered
- Weak assumption(B)/guarantee (H) pairs for specific context alternatives
Multiple Contexts Support

- The conjuncted contract format:
  \(<A, G, \{<B_1, H_1>, ..., <B_n, H_n>\}>\)

- Set of all possible contexts (correct environments)
  - Bounded by *the strong contracts*

- Subsets of the set of all possible contexts
  - Bounded by *the weak contracts*
  - Offer additional information on the behaviour of the component in the specific contexts
Case study

- **Wheel Braking System – WBS (by ARP4761, appendix L)**
  - Safety requirement: No single failure within the BSCU shall lead to “inadvertent braking due to BSCU”
  - Timing requirement: WBS execution is no more than 10ms
  - Previous work by [Damm et al] and [Cimatti, Tonetta]
Contracts for WBS (1)

**A**: Pedal1==Pedal2  
**G**: -

\{<B1: Platform=x and Compiler=y;  
H1: Delay between (Change(Pedal1,Pedal2),Change(Brake Line)) ≤10ms>;\}

**A**: Pedal1==Pedal2  
**G**: -

\{<B1: (SubBSCU1.Valid or SubBSCU2.Valid);  
H1: BSCU.Valid >;  
<B2: Platform=x and Compiler=y;  
H2: Delay between (Change(Pedal1,Pedal2), Change(CMD AS,AS)) ≤5ms>;\}

**A**: Pedal1==Pedal2  
**G**: -

\{<B1: Platform=x and Compiler=y;  
H1: Delay between (Change(Valid), Change(Brake Line)) ≤ 5ms>;\}

**A**: -;  
**G**: always terminates;

\{<B1: (Platform=x and Compiler=y) AND Valid1==TRUE;  
H1: Delay between (Change(CMD AS1,AS1), Change(CMD AS,AS)) ≤ 0,25ms>;  
<B2: (Platform=x and Compiler=y) AND Valid1==FALSE;  
H2: Delay between (Change(CMD AS1,AS1), Change(CMD AS,AS)) ≤ 1ms>;\}
Contracts for WBS (2)

A: Pedal1==Pedal2
G:-

{<B1: no fault in Monitor;
H1: SubBSCUi.Valid >;
<B2: (Monitor developed to DAL A);
H2: SubBSCUi.Valid with high confidence >;
<B3: Platform==x and Compiler==y;
H3: Delay between (Change(Pedal1, Pedal2),
Change(Valid,CMD AS,AS)) ≤ 4ms>;

A: Pedal1==Pedal2
G:-

{<B1: Platform==x and Compiler==y;
H1: Delay between (Change(Pedal1, Pedal2),
Change(Valid)) ≤ 2ms >;}
Conclusions and Future Work

- Just as components need to be designed for reuse, so do contracts as well.
- Strong and weak contracts are designed to support out-of-context development and reuse of components.
- The strong contracts bound the set of all possible contexts, while the weak contracts offer more information for specific subsets of that set.

Future work:
- Define relationship with safety argumentation.
- Extension of one of the contract languages with the strong and weak contract formalism.
Thank you!

Questions and comments?