Safety Contract Based Design (SCBD)

Andreas Söderberg (andreas.soderberg@sp.se) (+46 10 516 55 75)

05-11-2013
WoSoCER

This work has been funded by the research project nSafeCer within the ARTEMIS JU programme under grant agreement no 269265 and VINNOVA - Swedish Governmental Agency for Innovation Systems.
CONTENTS

• BACKGROUND
• SAFETY CONTRACTS
• SAFETY CONTRACT FORMULATION
• SIMPLIFIED EXAMPLE
• SAFETY CONTRACT VERIFICATION
• How ensure the validity of a safety contract to a third party SW-component user?
• Ongoing work
Background: Safety requirements contra functional/no-functional design requirements

Possible product development process:
1. Business case
2. Use case(s)
3. Functional design reqs.
5. Technical design reqs.

Further refinement.
Background: Contract-Based Design

- CBD (Contract-Based Design) is a well-known software engineering methodology, which is used for supporting development of component-based software systems (hereinafter called software contracts).

- Software contracts are used for defining/supervising the correlation between components in a system hierarchically.

- Software contracts contain pre-conditions and post-conditions (and in some cases maintenance-conditions) and acts as a native static-analysis tool, e.g. in certain programming languages (e.g. ADA 2012).

- What is common to all types of software contracts is that none of them contributes to the actual software system functionality, i.e. they may (should?) be removed when the software system is finalized.
Safety contracts

- The main difference between a safety contract and a software contract is that the software contract guarantees **functional properties** for a component interfaces/ports depending on its environment, in difference the safety contract guarantees a **integrity of safety constraints** depending on its environment (e.g. ASIL, SIL or PL)

- The software contract based design is a design method for supporting **integration testing** of a component based software system, the safety contract based design is a design method for supporting **certification** of such a system

- Both the guarantees and assumptions of a safety contract are comprised by **Safety Constraints**.

- Safety contracts may provide guarantees without any assumptions

- Safety contracts considers metadata only (e.g. development process used, amount of verification activities etc.) and **not** actual data as software contracts (e.g. assume input value type = int, guarantee returned value type = int)
A general component meta-model including safety contracts
Format of a safety contract

- For each and every guarantee in a safety contract, a series of assumptions is stated. These assumptions can be logically combined with the following two operators: [AND, OR]

- In this study we call such a combination of assumes and one guarantee as a safety contract fragment according to:

  \[A_1: \text{Safety Constraint 1}, \ [\text{AND, OR}] \]
  \[A_n: \text{Safety Constraint } n, \]
  \[G: \text{Safety Constraint} \]

- Each safety contract may contain an arbitrary amount of safety contract fragments

- Each safety constraint contains: One failure mode and the safety integrity level for avoiding this failure mode
Format of a safety contract, cont’d

The safety constraints of a safety contract should be formulated in a consistent and stringent manner by using **semi-formal notation**, using guide words, e.g.:

- Commission
- Omission
- TooHigh/TooLow
- TooLate/TooEarly
- etc.

- Interference w.r.t. execution time/memory
- Lack of resources
- Failure in platform service
Example of a safety contract for a program module

Safety contract fragment 1:
A1.1: CAN_frame, corruption, ASIL B AND
A1.2: CAN_frame.state, commission, ASIL C AND
A1.3: CAN_frame, TooLate, ASIL QM

G1: actuation, commission, ASIL C

Guidewords are strongly defined by the software components functional specification
Vertical and horizontal safety contract verification

Component decomposition

Application abstraction layer

Software component (SWC)

Service abstraction layer

Service components

Hardware abstraction layer

Hardware drivers

SC – Safety contract

SP Technical Research Institute of Sweden
Validity of safety contracts

Before the user of the software component can trust the safety contract the provider must convince the user about its validity with respect to the safety requirements specification (SRS) for the software component. This is achieved by using a Certification Authority (CA) who is a third and independent part.

The provider supplies the CA with all the necessary documents (verification tests performed etc.) including the safety contract and the source code for the software component (artifacts) for functional safety assessment (FSA).

When the FSA is finalized the CA produces a signature for the safety contract that covers all artifacts using a key not known by the provider.

Hashing could be used for generating the signature (e.g. CRC or MD5 signatures).
**Ongoing work**

- Currently in the project nSafeCer we are developing a tool that handles this type of safety contracts and can verify safety contracts (WP200)

- In addition, we will utilize this tool and these principles for one of the nSafeCer demonstrator (WP500) which is an implementation of an AUTOSAR BSW (diagnostic module)

- In order to make the principle of safety contract based design useful the safety constraints have to be very carefully formulated and selected. NOT NEW, also applied in any safety requirements formulation. Bad formulation – explosion of lower lever detailed safety requirements
Questions ?