Requirements engineering

Gábor Árpád Németh

From the brainstorming to the prototype

Requirements engineering in a green field investment

Requirements engineering in a green field investment 8/1
From the brainstorming to the prototype

1. Initial thoughts - Brainstorming
   - What we would like to achieve?
   - Define scope
   - Define possible customers
   - Define initial budget (and maybe some forecast for later stages)

2. Refine requirements
   - Risks:

Too general
→ too complex architecture → too complicated to implement a simple functionality
→ most of the possible functionalities will never be used

vs.

Too specific
→ bad architecture → dirty hacks over hardwired structure → hard to maintain
→ no future-proof (lack of important features)

Requirements engineering in a green field investment 8/2
From the brainstorming to the prototype

2. Refine requirements

- We should get known possible customers, their goals, problems and current processes…etc.
  - Otherwise: Ivory tower: we make something that nobody really wants…

An example: Ikarus PALT*:

A big engineering achievement that was not applicable to the existing infrastructure

Source of figures: [Ikarus archives]   *PALT: Passengers And Luggage Together

Requirements engineering in a green field investment 8/3
From the brainstorming to the prototype

- An example of collecting requirements – in a structured document:

Requirement Specification of Tool Zebra

Contents

1.2 Terminology

1.2.1 Behavior

The behavior is a dynamic description. It describes the interaction between the different participants of the system.

1.2.2 Topology

The topology is a static description, which defines the participants of the system and the interconnections among them. The parts of the Topology are described in Figure 1.

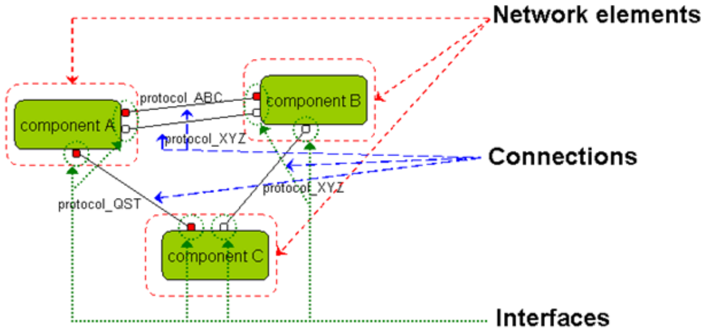


Figure 1: Parts of the topology

1.2.2.1 Network Elements

- The Network Elements are the nodes of the Topology. They are the participants of the interaction in the Topology.

R_G_20 **Handle different abstraction levels.**

One of the best properties of the engineer is the ability to select the right amount of details for a given task. It means that (s)he can consider a higher abstraction level if required, and not confused by any unimportant details. Thus, the tool should support the visualization and the editing in different abstraction levels. The tool should give the opportunity to the user to fold and unfold the given abstraction level on demand, to hide the details, which he is not interested in at a given time.

There are many requirements, which are related to this high-level requirement, see requirements R_G_21, R_U_05, R_U_06, R_T_03 and R_T_04 for further details.

R_G_21 **Handle different abstraction levels of the topology**. The tool should support to fold and unfold the different parts of the topology on demand.

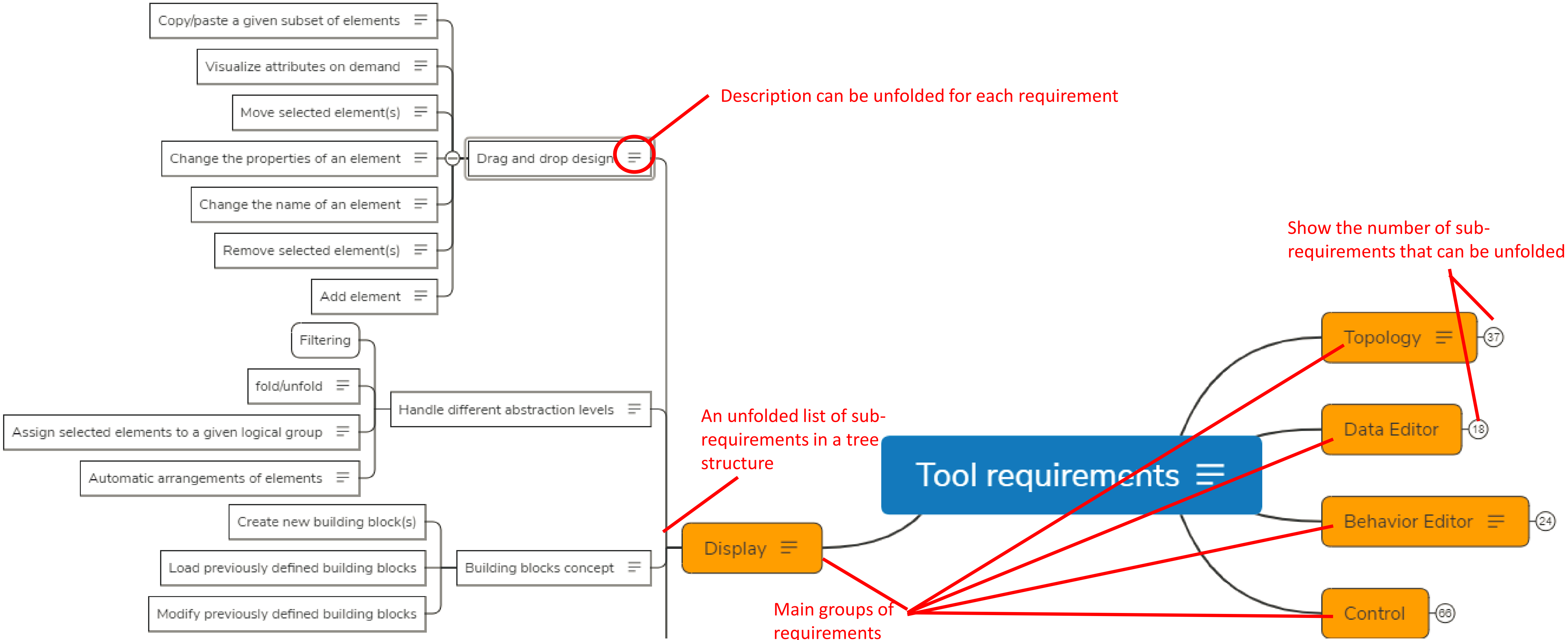
The user wants to display only that part of the network, which he is interested in. He wants to create higher and lower level view on demand.

An implementation idea would be if

(1) the user double clicks to an item, which he

Requirements engineering in a green field investment 8/4
From the brainstorming to the prototype

- An example of collecting requirements – tree structure (xmind)



Requirements engineering in a green field investment 8/5
From the brainstorming to the prototype

3. Create a prototype
   - **Only for proof of concept!**

   - Should answer the following questions (2/1):
     - **What** would we like to achieve?
       - List of functionalities
     - **How** would we like to achieve the goal? – non-functional aspects
       - Usability ↔ user interface, assumptions about users, working process
       - Performance related aspects:
         - Responsibility
         - Designed workload
         - Scalability
         - How to handle overload…etc.

         ↔ software architecture & required hardware
       - Security aspects ↔ architecture

Requirements engineering in a green field investment 8/6
From the brainstorming to the prototype

3. Create a prototype
   - Should answer the following questions (2/2):
     - How we should provide expected **quality**?
       - Manual testing for explanatory testing
       - A few proof-of-concept tests
         - Unit, integration, system levels
         - Functional and non-functional (performance, (G)UI, security…etc.)

Requirements engineering in a green field investment 8/7
From the brainstorming to the prototype

- Prototype:
  - **Role: Proof-of-concept**
  - Not an implementation code base for the final product!

- From most of the prototypes no real product has been developed due to the following reasons:
  - Wrong assumptions, when defining requirements and scope
  - Wrong initial thoughts about possible customers and/or their needs
  - Wrong assumptions about budget
  - The protype showed that the development cost and/or time would be too high
  - Organizational changes in the company resulted in cost cut / project closure
  - Similar product has been developed meanwhile in parallel

Requirements engineering in a green field investment 8/8
From the brainstorming to the prototype

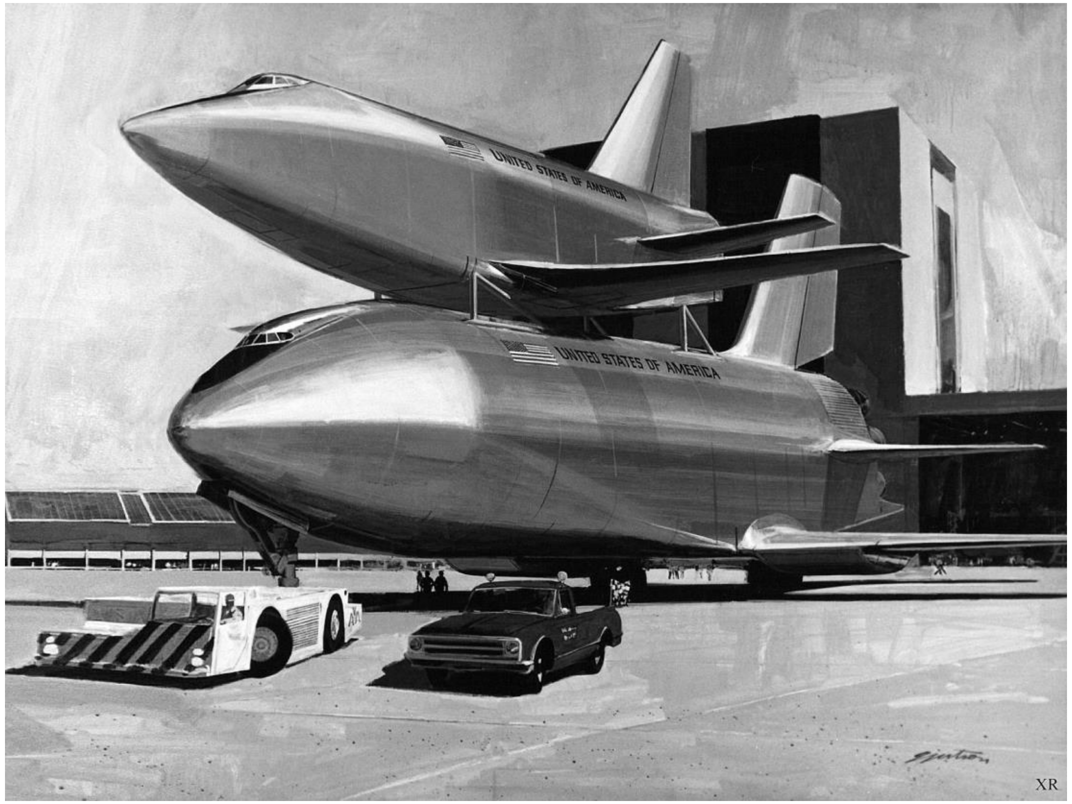
If we succeed than comes…

4. Productification
   - But many reviews before this step:
     - Technical reviews (architecture)
     - Management reviews at different levels (financial, customer…etc. aspects)

Project vs. product

	Project	**Product**
Generic	Unique, customer specific	Generic
Time	Has beginning and end date	Permanent (until phase out)
Planning	One-step/Predictive planning	Iterative/adaptive planning
Input	Project requirements	Evolving customer needs

A case study about changing requirements 9/1



The true story of the genesis of the Space Shuttle

References:

- David Baker: NASA Space Shuttle. 1981 onwards (all models). Owner's Workshop Manual. Haynes. 2011.
- Wikipedia: Space Shuttle program, Space Shuttle design process, Criticism of the Space Shuttle program

A case study about changing requirements 9/2

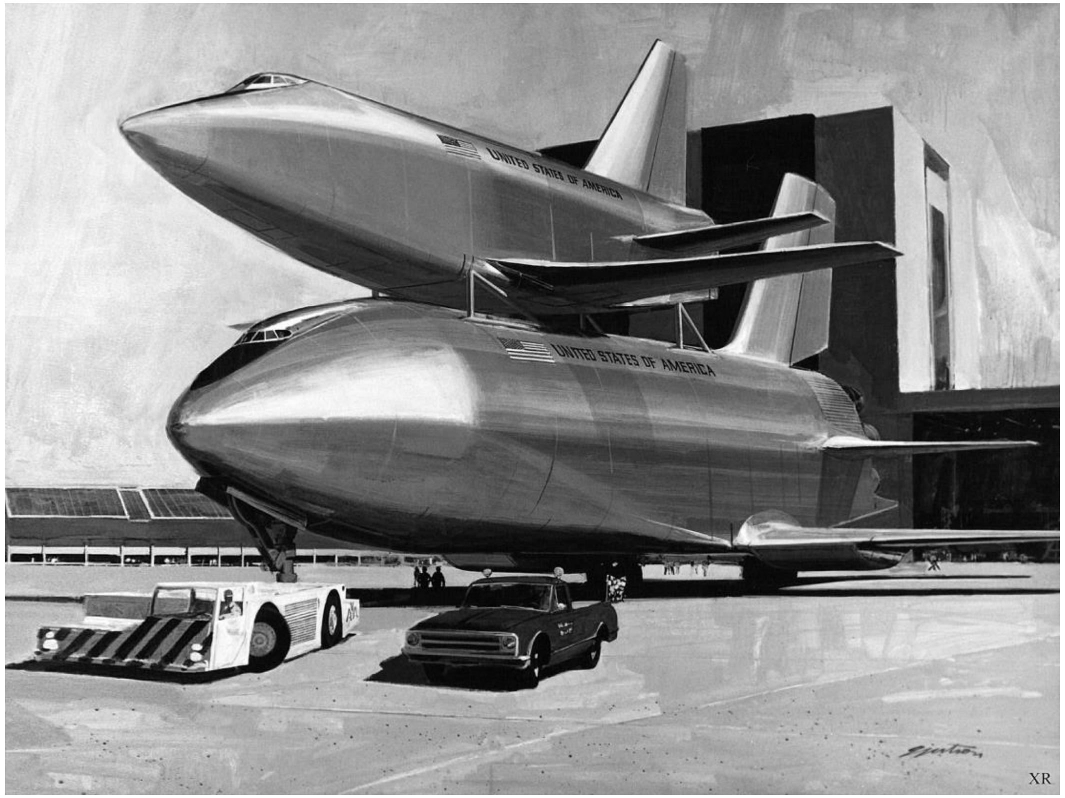
Background:

- After the Apollo (Moon landing program), significant cut on NASA's budget
- NASA plan to develop a fully reusable system – a „Shuttle" – to make space travelling significantly cheaper

Initial design (1970 Phase B studies):

- Fully reusable system:
  - 1.700 ton fly back manned booster with 12 rocket engines (with liquid fuel) and wings
  - 380 ton orbiter with 2 rocket engines
  - Orbiter's Payload capacity: 11 tons to LEO*

    * LEO: Low-Earth Orbit
- This solution was too costly to develop…



A case study about changing requirements 9/3

Investigate different concepts:

1. Fully (booster + orbiter) reusable systems
2. Expandable tanks
3. Expandable boosters

...etc

Catch-22:

- Lowest cost-per-flight solutions requires highest development cost
    - NASA have insufficient budget
- If decrease development cost, it results in a higher cost-per-flight
    - Controversial to the initial goal

A case study about changing requirements 9/4

Optimizations:

1. Expandable tank has been selected to decrease development cost (smaller orbiter would be enough)

   → compromise: not fully reusable system

2. SRBs (solid rocket boosters) proposed instead of liquid propellant ones to decrease cost-per-flight
   - Advantages:
     - **Simple and cheap**
     - Much easier to handle, no fueling needs before launch
   - Disadvantage:
     - Less efficient than liquid propellant rockets
     - **Once ignited, can not be stopped** – 1st compromise on safety

     (NASA had a rule before to not use them for manned space flights)

3. Insufficient thrust can be gained from solid rockets to lift-off the entire system

   → instead of the usual serial concept, parallel concept is selected for solid rockets

4. Recoverable boosters proposed to decrease cost-per-flight

A case study about changing requirements 9/5

Wait, initial design has been altered!

Do we need the cheaper solid rocket boosters if they are reusable?

A case study about changing requirements 9/6

Compromise of partners / payload capacity:

- The development cost was still too high → NASA found a partner (USAF*) to share the costs
- What payload is required?**
  - NASA: 6,8 tons to LEO*** (for satellites)
  - USAF*: 18 tons to polar orbit ≈ 30 tons LEO*** (for military satellites)
  - NASA later: 20 tons to LEO*** (to build Freedom space station from modules)

* USAF: US Air Force

** **The most important question when designing the Space Shuttle…**

*** LEO: Low-Earth Orbit

A case study about changing requirements 9/7

Canceled Safety functions:

- Liftoff – designed due SRBs:
  - Blow out port for boosters to separate in case of failure during ascending

    → cancelled due weight
  - Abort solid rocket motors

    → cancelled due simplicity

- Landing:
  - Turbofan engines that pop out from a compartment of rear payload bay at landing

    → cancelled due weight and volume

A case study about changing requirements 9/8

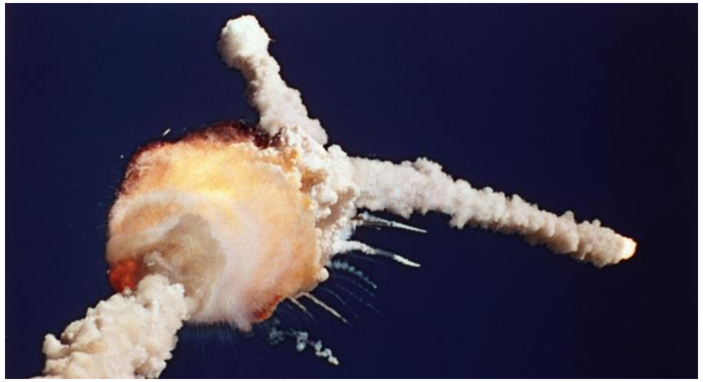
The usage of Space Shuttle - facts:

1. Due SRBs and parallel design, two Space Shuttle disasters:
   - Challenger in 1986

   → decrease flight intensity

   → US Air Force back out from project

   → increase cost-per-flight



   - Columbia in 2003

   → increase cost-per-flight

   → decision about the retirement of the Space Shuttle fleet

A case study about changing requirements 9/9

2. Except HST* in 1990, only after 1998 (17 years after first flight!) NASA uses the possible payload capability of the Space Shuttle, when building the ISS**

   * HST: Hubble Space Telescope

   ** ISS: International Space Station

3. The Vandenberg Space Shuttle launch pad build for US Air Force has never been used

4. The initial plan to send Space Shuttle into space bi-weekly has been never achieved

   → the cost-per-flight has been even higher compared to simple, expandable rockets!

5. After Space Shuttle era:
   - More simple designs
   - Concentrate on liquid propellant rocket engines
   - Concentrate on less payload capacity

From the customer requirements to the specification

Requirements engineering in brown field investments

Requirements engineering in brown field investments 11/1

From the customer requirements to the specification through an example process

An overview:

1. CR from customer
2. Early estimation
3. Task clarification → Feature Specification
4. Design documents (architectural, test…etc.)
5. Implementation
6. Tests
7. Documentation
8. Deployment

Focus on this topic

Scheduling of CR at any stage is made by PO according to priorities / available resources / output of previous stages

CR: Change Request
PO: Product Owner

Requirements engineering in brown field investments 11/2

From the customer requirements to the specification through an example process

1. Customer requests a change
   - Submits a CR (change request) into a CR management system (example tool: Tuleap)
   - Describes the requested functionality from the customer perspective
     - May be ambiguous, may not be self consistent, may lack of important details…etc.

<table>
<tr><th>Artifact number</th><th>Title</th><th>Customer</th></tr>
<tr><td>Artf010416</td><td>Efficient CDA handling in CoT</td><td>XYZ</td></tr>
<tr><td colspan="3">CoT should handle CDA:
A proper mechanism need to be implemented for D-INVITE, message exchange regarding DSoP, and initiating PO. This mechanism should be implemented between HEs of LoSP.
Note that a WO handover is also necessary.</td></tr>
</table>

Requirements engineering in brown field investments 11/3

From the customer requirements to the specification through an example process

2. Based on the CR an early estimation is made
   - By a business analyst/requirement engineer/system architect
   - Quickly with limited efforts
   - Output:
     - A quick overview of the topic, affected part(s) of the system, possible bigger tasks
     - Polo size: S/M/L/XL

       → determines the rough timeframe in mhrs* required for development, tests, documentation and deployment

       → each domain/company/company units may have different timeframes for each polo size

Polo size	mhrs
S	0-40
M	41-80
L	81-200
XL	200+

Polo size	mhrs
S	0-200
M	200-500
L	500-2000
XL	2000+

* mhrs: men hours

Requirements engineering in brown field investments 11/4

From the customer requirements to the specification through an example process

3. Task clarification with customer
    - Iterative process
    - Transparency - CR management system:
        - The communication should be tracked
            - To avoid later misunderstandings
            - To provide the ability to involve new people from both sides
        - The status of the CR should be updated
    - Always check related standards!
        - Conformance to related standards is important
        - If we must deviate from the standard, then write down the reason behind it & the possible risks
    - Always checks related existing features!
        - Backward compatibility is important

CR: Change Request

Requirements engineering in brown field investments 11/5

From the customer requirements to the specification through an example process

3. Task clarification with customer

- Output: [Feature Specification]
  - Describes the required functionality in an unambiguous, self-consistent way that can be given to the developers/testers/technical writers

Artf010416: Efficient CDA handling in CoT

Keyword(s)

D-INVITE, CoT, DSoP, WO handover

Abstract

This feature specification is intended to be an agreement between the different CoT of OWU on the detailed requirements related to CDA.

Contents

1 Introduction

1.1 Statement of problem

Nowadays, the CDA of CoT is a more and more pronounced problem that has to be solved with a proper method. The D-INVITE, the message exchange regarding DSoP and PO process should be implemented in the CoT both for W and O legs.

1.2.2 Standards, Specifications and Studies

Title	Number	Revision	Comment
[I] A Standard for the Transmission of IP Datagrams on Avian Carriers	RFC 1149	-	
[II] IP over Avian Carriers with Quality of Service	RFC 2549	-	An enhanced version of RFC1149

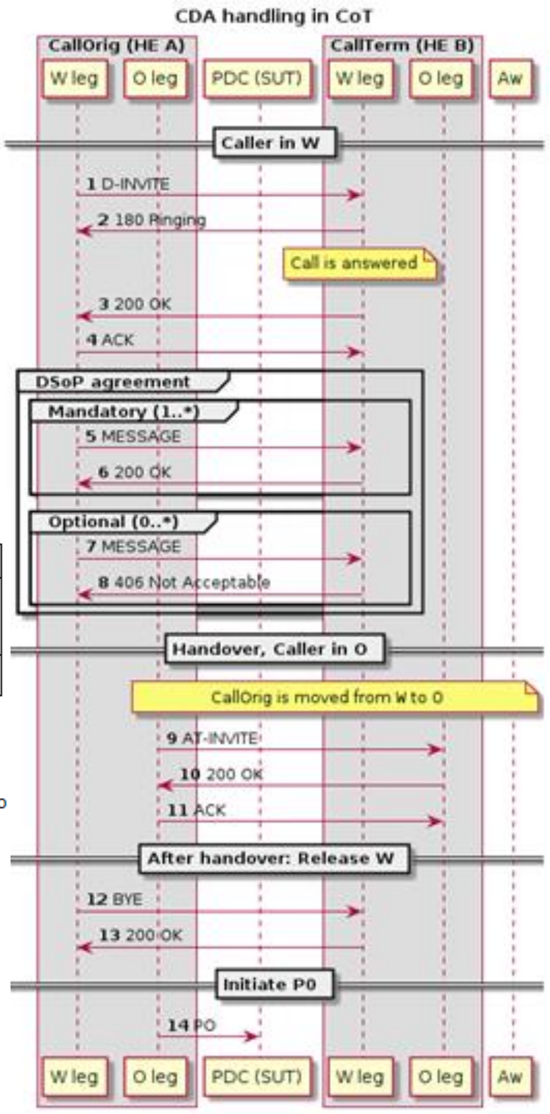
1.3 Scope

The requested feature is the interoperability between the HEs of CoT to perform CDA with DSoP.
The CDA functionality enables handover from a W network to legacy O networks and a fallback from O to W before W leg is released (note that no fallback opportunity is required after the W leg is released).

1.4 Abbreviations

CDA	Common DA
CoT	Colleagues of TST
CFR	Chinese Food Restaurant
DA	Dinner Arrangement
DB	DataBase
DSoP	Detailed Specification of Pizza

2.1.1.1.3 Call Flow



2.2 System Impacts

It is definitely necessary to write an IP for this feature.

For W leg either whitespace [3], gothic [1] or usual character coding [2] can be used.

Please also take attention to create the *CDA handling in CoT* part of TitanSim Online help; significant documentation work is required.

2.3 Test Analysis

The feature cannot be tested in lab1 due to lack of CDA functionality, it can be tested only in CoT environment.

Selftest must to be written for the use cases described in this document, and in the future it is advised to create test also in ONTE.

However, the power of manual testing should not be underestimated, even the postcondition of PO process should be tested.

2.4 Non-Function Requirements

2.4.1 Capacity and Performance

No request has been received

Significant performance drop is expected if W legs of HE A and B are implemented over [I] or [II] instead of legacy IP networks.

2.5 Customer Impacts

No customer impacts on existing functionalities.

2.6 Backwards Compatibility

No backward incompatible issue is foreseen.

2.7 Technical Risks

Problems may occur in the interpretation of the received message is expected if whitespace [3] character coding is used over [I] or [II]. In this case the using of other character coding methods or transport layers is proposed.

Requirements engineering in brown field investments 11/6

From the customer requirements to the specification through an example process

3. Task clarification with customer
   - Output: Feature Specification
     - Describes the required functionality in an unambiguous, self-consistent way that can be given to the developers/testers/technical writers
       - classified into use-cases or user stories

Use cases:

- *Business artefacts* defining some software requirements.
- Describes the actions or steps of events:
  - *`Precondition`*
  - *`Action 1`*
  - *`Action 2`*
  - *`Postcondition`*

User stories:

- Short, simple descriptions of a feature told *from the perspective of the customer*.
- They typically follow a simple template:
  `As a <type of user>, I want <some goal> so that <some reason>.`

Requirements engineering in brown field investments 11/7

From the customer requirements to the specification through an example process

3. Task clarification with customer
    - Output: Feature Specification
        - Describes the required functionality in an unambiguous, self-consistent way that can be given to the developers/testers/technical writers
            - classified into use-cases or user stories
            - A part of it may contain formal descriptions *(like the message sequence chart in the figure)*
            - Should be self consistent (provide used abbreviations, references…etc.)
            - Should contain information about risks
            - May contain information about test design

Requirements engineering in brown field investments 11/7

From the customer requirements to the specification through an example process

Standards

- Established norm or requirement in regard to technical systems
- Formal document that establishes uniform engineering or technical criteria, methods, processes, and practices

- Examples:
  - An RFC standard: RFC 3261 SIP: Session Initiation Protocol
  - A 3GPP standard: 32.299 Diameter protocol, charging management

Requirements engineering in brown field investments 11/9

From the customer requirements to the specification through an example process

3. Task clarification with customer
    - Output: Feature Specification
        - Must be accepted by both sides:
            1. Reviewed internally
                - Participants:
                    - Business analysts/system architects
                    - Developers (who have competence in the related part of the software)
                    - Test responsible person
                - Review responsible:
                    - Screening
                    - Moderate review, give verdict (accepted / accepted with comments / rejected)
                    - Check afterlife based on verdict (check modifications to comments / 2nd turn of review…etc.)
                    - Update status on CR management system
            2. Approved by customer

Requirements engineering in brown field investments 11/10

From the customer requirements to the specification through an example process

Possible risks:

- We want that feature right now!
  → Hardwired, too specific solutions that are hard to be generalized or maintain
- Give too big requirements at one step without priorities and schedule
  → Will be never finished
- Requirements that do not conform with corresponding standards
  → Compatibility problems at later phase, working mode-switch and other dirty hacks

Requirements engineering in brown field investments 11/11

From the customer requirements to the specification through an example process

Possible risks:

- Problems with documentation
  1. No proper documentation of task clarification discussions with user

     → misunderstandings at deployment, blaming each other

     → changing requirements

     → delay of delivery, more cost effect

  2. No proper documentation of the delivered feature

     (missing or incomplete user / developer / architectural documentations)

     → customer/ developer unable to use the feature properly, requirements and design decisions are mixed up

     → reverse engineering (in code / standards / old e-mail exchanges with customer)

     → try to sort related documents out and get approval by customer

     → huge additional costs, loss of credibility

  3. No traceability exists between specification – code – test – documentation 4-tuple

Feature group	Feature name	Uniq ID	Role	Subfeatures (if exist)	Related parameters	Related Standard	Link to user help	Feature Specification	Related tests