Institute of Software Technology
Reliable Software Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Software Performance: Altering Capitalist Theory and Symbol

Max Musterman

Course of Study:	Softwaretechnik
Examiner:	Dr.-Ing. André van Hoorn (Prof.-Vertr.)
Supervisor:	Dr. Dušan Okanović, Teerat Pitakrat, M.Sc.
Commenced:	Juli 5, 2013
Completed:	Januar 5, 2014
CR-Classification:	I.7.2

Abstract

... Short summary of the thesis in English ...

Kurzfassung

... Short summary of the thesis in German ...

Contents

List of Figures

List of Tables

List of Acronyms

FR	Fehlerrate

List of Listings

List of Algorithms

Chapter 1

Introduction

In diesem Kapitel steht die Einleitung zu dieser Arbeit. Sie soll nur als Beispiel dienen und hat nichts mit dem Buch [**WSPA**] zu tun. Nun viel Erfolg bei der Arbeit!

Bei LaTeX werden Absätze durch freie Zeilen angegeben. Da die Arbeit über ein Versionskontrollsystem versioniert wird, ist es sinnvoll, pro *Satz* eine neue Zeile im `.tex`-Dokument anzufangen. So kann einfacher ein Vergleich von Versionsständen vorgenommen werden.

Thesis Structure

Die Arbeit ist in folgender Weise gegliedert:

Kapitel ?? – ??: Hier werden werden die Grundlagen dieser Arbeit beschrieben.

Kapitel 4 – Conclusion fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

Chapter 2

The On-board Software Reference Architecture (OSRA)

Introduction

Background

Space industry has recognized already for quite some time the need to raise the level of standardisation in the avionics system in order to increase the efficiency and reduce cost and schedule in the development. The implementation of such a vision is expected to provide benefits for all the stake-holders in the space community:

Customer Agencies Significant drop in the project development lifecycle and the risk involved in the software development

System Integrators There would be increased competition amongst them to deliver at lower price and maintain shorter time-to market and there would also be multi-supplier option

Supplier Industry Benefits from diversified customer bases and the supplied building blocks would be compatible with prime architectures across the board

Similar initiatives have already been taken across various industries and eg. AUTOSAR for the automotive industry is worthy mentioning. Space can benefit from these examples by studies related to how these similar initiatives were successfully conducted and how they fared. ALthough the business model is different in the automotive and the space sectors, AUTOSAR demonstrates the need fir standardization is the key irrespective of the sector and is driven by the need of the industry to become more competitive.

Space primes and on-board software companies have made significant progress and have implemented and/or are implementing reuse on the basis of company's internal software refernce architectures and building blocks. However in for this standardization to provide maximum benefits, it has to be tackled at the European level rather than at company level.

ESA through its two parallel activities aimed at increasing the software reuse in on-board softwares (CORDET and Domeng) have confirmed that interface standardization allows to efficiently compose the software on the basis of existing and mature building blocks.

To refer to all ongoing initiatives and to provide a platform for technical discussions, related to the vision of avionics development through maximizing reuse and standardization, a "Space Avionics Open Interface Architecture" Advisory Group (SAVOIR Advisory Group) was created. SAVIOR Advisory Group decided to spawn a specific subgroup on-board software reference architectures called "SAVOIR Fair Architecture and Interface Reference Elaboration" working group (SAVOIR FAIRE). OSRA is the result of R&D activities of this group.

The On-board software reference architecture (OSRA) is designed to be a single, common and agreed framework for the definition of the on-board software (OBSW) of the future European Space Agency (ESA) missions. It is based on solid scientific foundations and accompanied by development methodology and architectural practices that fit the domain. A single software system would thus be an "instantiation" of the reference architecture to specific mission needs.

The software architecture is the key to create "good quality" software because it promotes architectural best practices and contributes to the quality of the software. A bad architecture hinders the fulfillment of functional, behavioral, non-functional and life-cycle requirements. Elevating a software architecture to software reference architecture permits to gather and re-use lessons learned and architectural best practices, give new projects a consolidated running start and promote a product line approach.

Need for reference architecture

Motivation

The schedules of space projects are always decreasing and the team need to increase their efficiency and cost effectiveness in the development process of on-board avionics. But the on-board software is getting more complex because of the trend towards more functionality being implemented by the on-board software. Therefore the overall objective of space industry is now to standardize the avionics systems and therefore the on-board software.

A building block approach is one of the ways to tackle this problem. In this approach, the on-board software is implemented from a set of pre-developed and fully compatible building blocks, plus specific adaptations and "missionisation" according to specific mission requirements. The target missions are the core ESA missions, ie.e high reliability and availability spacecraft driven systems (eg. operational missions, science missions).

The "right" building blocks need to be produced and supplied by the suppliers to any system integrator and to achieve this, reference architectures need to be defined

Separation of the application aspects from the general-purpose data processing aspects is the key to generic/reusable software architectures. The lower layers of the architectures usually handle the implementation of communication, real time capabilities etc and the higher level layers usually deal with the application aspects. However there have to be ways to annotate the application building blocks (ABB) with sufficient information regarding requirements related to communication, real-time, dependability etc., so that the platform building blocks (PBB) can provide the suitable complete implementation. Development of interface specifications with reference architectures as the basis allows the implementation of the famous AUTOSAR concept: "Cooperate on standards, compete on implementation"

User needs

These are some of the needs that were assimilated to guide the development of the software reference architecture:

Shorter software development time The software development schedule should be reduced because usually the definition of the software requirements is done at a later stage and the final version of the software is expected to be released earlier. Even though the cost of the software itself us a minor fraction of the cost of the whole system in space industry, the impact of delays in availability of the software may have a huge impact on the overall schedule and consequently on the cost of the project

Reduce recurring costs It is important to identify and reduce the recurring costs and in turn help to use the project resources to focus on value added to the product or to reduce the cost of development while providing the same set of functions. Examples for recurring costs include device drivers, real-time operating system, providing communication services etc and it is important to note that these cots drivers do not provide an added value and are not mission specific.

Quality of the product The level of the quality (timing predictability, dependability of the software etc) of the software must at least be the same as the one of OBSW developed with current approaches.

Increase cost-efficiency Cost-efficiency is the "value" of the software product that is developed with a certain amount of budget. An increased cost-efficiency is achieved by developing the same set of functions for less budget, developing the same set of functions with more stringent requirements for the same budget and increasing the number of realized functions for the same budget. The budget available for the software development is not expected to grow and it may be indeed be subjected to reduction and hence new development approaches may be required to fulfill this user need. On contrary the performance of the application building blocks eg. accuracy of the AOCS controls is expected to grow and new complex functionalities are expected to develop

Reduce Verfication and Validation effort The main contributor to the cost of software development are V&V activities which contribute anywhere between 50% to 70% of the overall cost. Adoption of the principle of Correction by Construction (C-by-C) which is one of the founding principles of choice (refer Chapter 2), analysis at early design stage and provisions for re-usability of (functional) tests are expected to reduce the Verification and Validation efforts. This also leads to shorter software development times and reduced costs.

Mitigate the impact of late requirement definition or change Late refinement of system design, evolution of the operational level, late finalization of the system FDIR, software modification to compensate the problems in the hardware found during system integration may often lead to definition of new requirements or their changes for the software, anytime in the entire SW life-cycle.

Support for various system integration strategies Preliminary software releases are important to allow early system integration and software development may be managed with different strategies. It is necessary to respect these strategies and help final integration of increments or elements.

Simplification and harmonization of FDIR Simplification and coordination of the Fault Detection, Isolation and Recovery (FDIR) needs to be handled at both the system and software level. System engineers and software implementers need to justify the definition of the FDIR strategy at the system level and the software level respectively. A set of functionalities and design patterns need to be provided at the software level that cater to necessary mechanisms for the software realization of FDIR strategy

Optimize flight maintenance Flight maintenance may be required to change the OBSW and provision of the required operations and coordination of the strategy to perform it will decrese the time and cost of maintenance. It is better if parts of the software could be updated without having to reboot the CDMU.

Industrial policy support The development process should enable multi-team software development. It is necessary to incorporate certain flexibility in the allocation of the software elements to industry, according to certain criterion such as prime/non-prime, or geographical return. Multi-team software development is essential to subcontract to non-primes while be in charge of the integration and apply the geographical return policy.

Role of software suppliers As discussed before, the new approach must increase the competence of the supplier and foster competition amongst the suppliers: Different suppliers may develop the same component and compete on quality, extension features, performance, cost and schedule. The suppliers will also profit from this approach as they do not have to adapt the software to specific development policies if each single prime.

Dissemination activities System engineers can be exposed to core principle of the process and if they derive specifications for the system out of the domain of reuse, the costs will certainly increase.

Future needs The trend of increasing complexity of the OBSW gives rise to several needs and these needs need to be subjected to evaluation and their impact on the software reference architecture needs to be monitored. Some of the examples of the future needs include integration of functions of different criticality and security levels, use of Time and Space Partitioning (TSP), support to the multi-core processors, contextual verification of safety properties.

High level requirements

The user needs were translated into a set of high-level requirements:

Software reuse The architecture shall be designed in such a way that the reuse of the functional aspects should be independent of the reuse of the non-functional aspects, reuse of the the unit, integration and validation tests by providing a pre-qualification data package supported by a SW Reuse File in the sense of ECSS software standards. Traced to user needs:

- Shorter software development time
- Reduce recurring costs

Separation of concerns Separation of concerns is one of the cornerstone principles and it deals wit separating different aspects of the software design, in particular the functional and non-functional concerns. Separation of concerns helps to reuse functional concerns independently from non-functional concerns and hence increasing the software reuse. Traced to user needs:

- Quality of the product
- Reduce Verification and Validation effort
- Role of software suppliers

Reuse of V&V tests The chosen architectural approach should also promote the reuse of Verification and Validation tests that were performed on the software and not just the software itself. The aim is to maximize the reuse of the tests written for the functional part of the component software. Traced to user needs:

- Shorter software development time
- Reduce Verification and Validation effort

HW/SW Independence It enables development of the software independent from the hardware features. Its is necessary to separate parts of the software that interact directly with the hardware into separate modules and make them accessible through defined interfaces. In this way, as long as the interface does not change, the software isolated from the changes in the hardware-dependent layer. Traced to user-needs:

- Quality of the product
- Mitigate the impact of later requirements definition or change
- Support for various system integration changes

Component based approach The whole software is designed as a composition of components that are reusable in nature. The architecture shall respect preservation of properties of individual building blocks once integrated into the architecture and it should be possible to calculate the system's property as a function of components' individual properties. The former is called composability and the latter is called compositionality. Traced to user needs:

- Shorter software development time
- Reduce recurring costs
- Increase cost-efficiency
- Support for various system-integration changes

- Product policy
- Role of software suppliers

Sofwtare observability The software architecture should provided means to observe the software specific parts and extract current and past status of the software using the services specified by its operational scenarios. This prevents the need for post launch updates or patches of the software in case of failure analysis needs. Traced to user needs:

- Quality of the product
- Reduce Verification and Validation effort
- Simplification and harmonization of FDIR
- Optimize flight maintenance

Software analysability he design process and methodology used for the reference architecture shall support the verification at design time of functional and non-functional properties. Traced to user needs:

- Quality of the product
- Reduce Verification and Validation effort

Property preservation The non-functional properties become the constraints on the system as they specify the "frame" in which the system is expected to behave and be consistent with what was predicted during the analysis. These properties have to be preserved or enforced so that these properties are not only used for the analysis of the software model, but also find their way through to the final system at run-time. Adequate mechanisms should be provided to handle the enforcement of properties and reactions to violation of the properties. Traced to user needs:

- Quality of the product
- Reduce Verification and Validation effort

Integration of software building blocks The architecture should allow the combination of coherent building blocks

- Shorter software development time
- Mitigate the impact of late requirement definition or change
- Support for various system integration strategies
- Product policy
- Role of software suppliers

Support for variability factors In order to reduce the complexity of the architecture, the potential variation of the architecture induced by the variation of the domain must be isolated in some places such as reuse is improved and need for modification is decreased. Traced to user needs:

- Increase cost-efficiency

Late incorporation of modification in the software The architecture should be immune to modification of the software late in the software life cycle. System integration always finds some system problems and it is the responsibility of the software to contain these problems and implement new requirements. The architecture to which the software is conformal to, should be able to handle these late modifications in the software.Traced to user needs:

- Mitigate the impact of late requirement definition or change

Provision for mechanisms for FDIR The aircraft dependability should be handled by the architecture and in particular the Fault Detection, Isolation and Recovery. Traced to user needs:

- Simplification and harmonization of FDIR

Sofwtare update at run-time The reference architecture should allow update to single software components as well as their bindings without having to reboot the entire on-board computer as it is a risk for the system and reduces the mission availability/uptime. Traced to user needs:

- Optimize flight maintenance

OSRA comprises of three layers: the component layer, the interaction layer and the execution platform `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Appendix A.

Chapter 3

Heading on level 0 (chapter)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. $\sin^2(\alpha) + \cos^2(\beta) = 1$. If you read this text, you will get no information $E = mc^2$. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. This text should contain all letters of the alphabet and it should be written in of the original language. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. There is no need for special contents, but the length of words should match the language. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$.

3.1. Heading on level 1 (section)

Hello, here is some text without a meaning. $\mathrm{d}\Omega = \sin\vartheta \mathrm{d}\vartheta \mathrm{d}\varphi$. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $\sin^2(\alpha) + \cos^2(\beta) = 1$. This text should contain all letters of the alphabet and it should be written in of the original language $E = mc^2$. There is no need for special contents, but the length of words should match the language. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$.

3.1.1. Heading on level 2 (subsection)

Hello, here is some text without a meaning. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. This text should show what a printed text will look like at this place. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$. If you read this text, you will get no information. $\mathrm{d}\Omega = \sin\vartheta \mathrm{d}\vartheta \mathrm{d}\varphi$. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special contents, but the length of words should match the language. $\sin^2(\alpha) + \cos^2(\beta) = 1$.

Heading on level 3 (subsubsection)

Hello, here is some text without a meaning $E = mc^2$. This text should show what a printed text will look like at this place. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. If you read this text, you will get no information. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$. This text should contain all letters of the alphabet and it should be written in of the original language. $\mathrm{d}\Omega = \sin\vartheta \mathrm{d}\vartheta \mathrm{d}\varphi$. There is no need for special contents, but the length of words should match the language.

Heading on level 4 (paragraph) Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. $\sin^2(\alpha) + \cos^2(\beta) = 1$. If you read this text, you will get no information $E = mc^2$. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. This text should contain all letters of the alphabet and it should be written in of the original language. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. There is no need for special contents, but the length of words should match the language. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$.

3.2. Lists

3.2.1. Example for list (itemize)

- First item in a list
- Second item in a list
- Third item in a list

- Fourth item in a list
- Fifth item in a list

Example for list (4*itemize)

- First item in a list
    - First item in a list
        - First item in a list
            - First item in a list
            - Second item in a list
        - Second item in a list
    - Second item in a list
- Second item in a list

3.2.2. Example for list (enumerate)

1. First item in a list
2. Second item in a list
3. Third item in a list
4. Fourth item in a list
5. Fifth item in a list

Example for list (4*enumerate)

1. First item in a list
    a) First item in a list
        i. First item in a list
            A. First item in a list
            B. Second item in a list

        ii. Second item in a list

    b) Second item in a list

2. Second item in a list

3.2.3. Example for list (description)

First item in a list

Second item in a list

Third item in a list

Fourth item in a list

Fifth item in a list

Example for list (4*description)

- **First** item in a list
  - **First** item in a list
    - **First** item in a list
      - **First** item in a list
      - **Second** item in a list
    - **Second** item in a list
  - **Second** item in a list
- **Second** item in a list

Chapter 4

Conclusion

Hier bitte einen kurzen Durchgang durch die Arbeit.

Future Work

...und anschließend einen Ausblick

Appendix A

LaTeX-Tipps

A.1. File-Encoding und Unterstützung von Umlauten

Die Vorlage wurde 2010 auf UTF-8 umgestellt. Alle neueren Editoren sollten damit keine Schwierigkeiten haben.

A.2. Zitate

Referenzen werden mittels `\cite[key]` gesetzt. Beispiel: [**WSPA**] oder mit Autorenangabe: **WSPA**.

Der folgende Satz demonstriert 1. die Großschreibung von Autorennamen am Satzanfang, 2. die richtige Zitation unter Verwendung von Autorennamen und der Referenz, 3. dass die Autorennamen ein Hyperlink auf das Literaturverzeichnis sind sowie 4. dass in dem Literaturverzeichnis der Namenspräfix “van der” von “Wil M. P. van der Aalst” steht. **RVvdA2016** präsentieren eine Studie über die Effektivität von Workflow-Management-Systemen.

Der folgende Satz demonstriert, dass man mittels `label` in einem Bibliopgrahie"=Eintrag den Textteil des generierten Labels überschreiben kann, aber das Jahr und die Eindeutigkeit noch von biber generiert wird. Die Apache ODE Engine [**ApacheODE**] ist eine Workflow-Maschine, die BPEL-Prozesse zuverlässig ausführt.

Wörter am besten mittels `\enquote{...}` “einschließen”, dann werden die richtigen Anführungszeichen verwendet.

Beim Erstellen der Bibtex-Datei wird empfohlen darauf zu achten, dass die DOI aufgeführt wird.

Listing A.1 lstlisting in einer Listings-Umgebung, damit das Listing durch Balken abgetrennt ist

```
<listing name="second sample">
  <content>not interesting</content>
</listing>
```

A.3. Mathematische Formeln

Mathematische Formeln kann man so setzen. `symbols-a4.pdf` (zu finden auf http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf) enthält eine Liste der unter LaTeX direkt verfügbaren Symbole. Z. B. $\mathbb{N}$ für die Menge der natürlichen Zahlen. Für eine vollständige Dokumentation für mathematischen Formelsatz sollte die Dokumentation zu `amsmath`, ftp://ftp.ams.org/pub/tex/doc/amsmath/ gelesen werden.

Folgende Gleichung erhält keine Nummer, da `\equation*` verwendet wurde.

$$x = y$$

Die Gleichung A.1 erhält eine Nummer:

$$x = y \tag{A.1}$$

Eine ausführliche Anleitung zum Mathematikmodus von LaTeX findet sich in http://www.ctan.org/tex-archive/help/Catalogue/entries/voss-mathmode.html.

A.4. Quellcode

Listing A.1 zeigt, wie man Programmlistings einbindet. Mittels `\lstinputlisting` kann man den Inhalt direkt aus Dateien lesen.

Quellcode im `<listing />` ist auch möglich.

A.5. Abbildungen

Die Figure A.1 und A.2 sind für das Verständnis dieses Dokuments wichtig. Im Anhang zeigt Figure A.4 on page 25 erneut die komplette Choreographie.

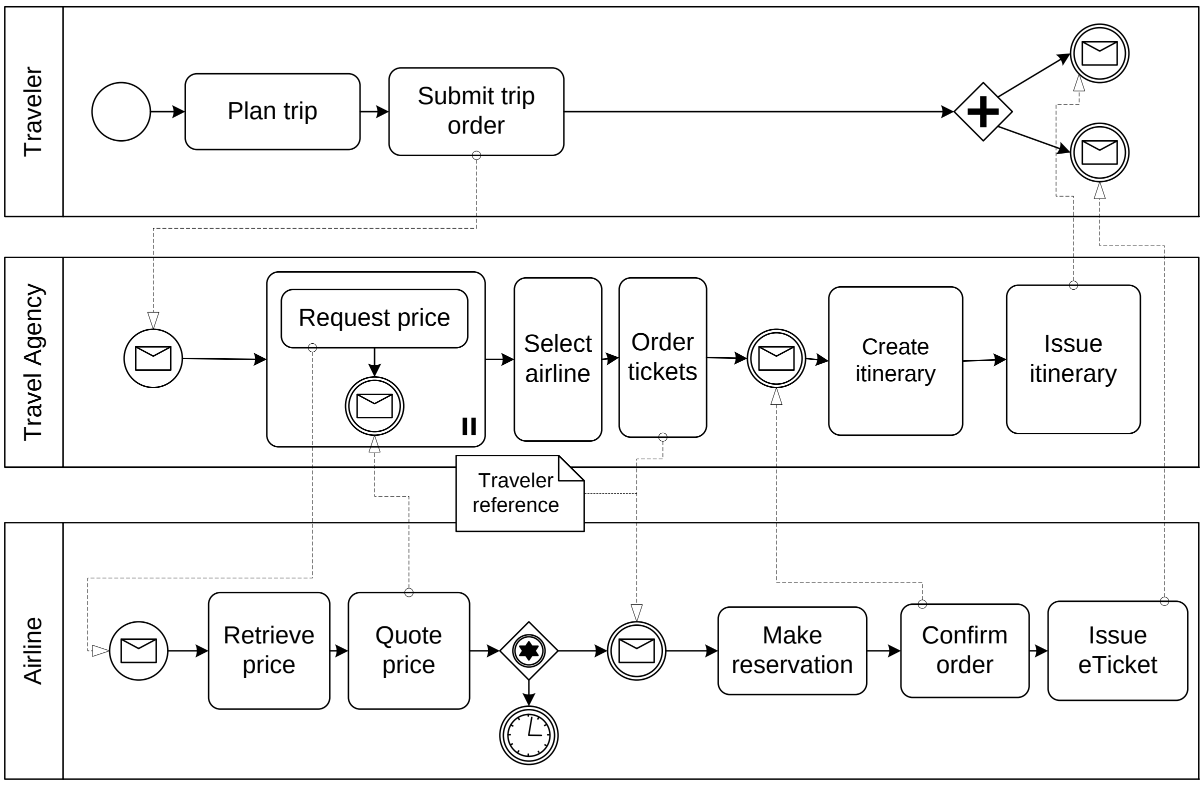

Figure A.1.: Beispiel-Choreographie

Das SVG in **??** ist direkt eingebunden, während der Text im SVG in **??** mittels pdflatex gesetzt ist.

Falls man die Graphiken sehen möchte, muss inkscape im PATH sein und im Tex-Quelltext `\iffalse` und `\iftrue` auskommentiert sein.

A.6. Tabellen

Table A.1 zeigt Ergebnisse und die Table A.1 zeigt wie numerische Daten in einer Tabelle representiert werden können.

A.7. Pseudocode

Algorithm A.1 zeigt einen Beispielalgorithmus.

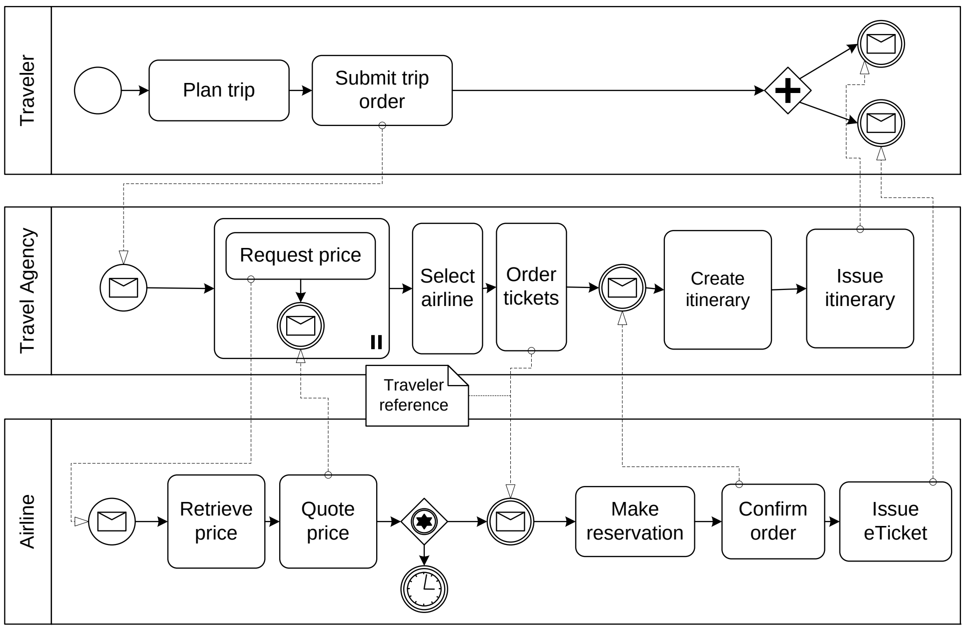

Figure A.2.: Die Beispiel-Choreographie. Nun etwas kleiner, damit `\textwidth` demonstriert wird. Und auch die Verwendung von alternativen Bildunterschriften für das Verzeichnis der Abbildungen. Letzteres ist allerdings nur Bedingt zu empfehlen, denn wer liest schon so viel Text unter einem Bild? Oder ist es einfach nur Stilsache?

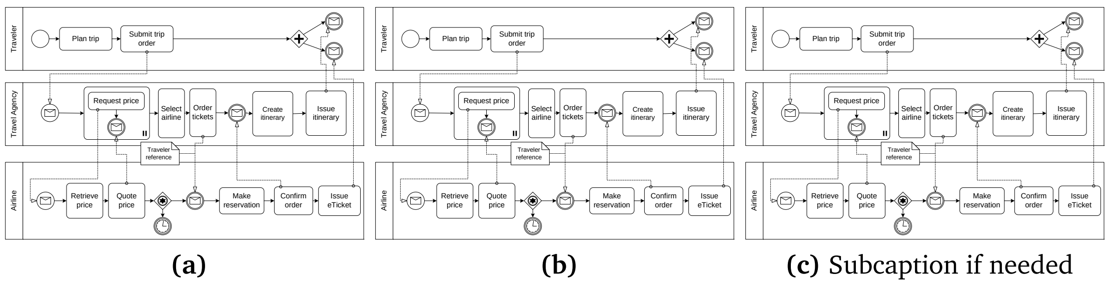
(a) **(b)** **(c)** Subcaption if needed

Figure A.3.: Beispiel um 3 Abbildung nebeneinader zu stellen nur jedes einzeln referenzieren zu können. Abbildung A.3b ist die mittlere Abbildung.

zusammengefasst		**Titel**
Tabelle	wie	in
tabsatz.pdf	empfohlen	gesetzt
Beispiel	ein schönes Beispiel für die Verwendung von “multirow”	

Table A.1.: Beispieltabelle – siehe http://www.ctan.org/tex-archive/info/german/tabsatz/

	Parameter 1		**Parameter 2**		**Parameter 3**		**Parameter 4**	
Bedingungen	**M**	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
W	1.1	5.55	6.66	.01				
X	22.22	0.0	77.5	.1				
Y	333.3	.1	11.11	.05				
Z	4444.44	77.77	14.06	.3				

Table A.2.: Beispieltabelle für 4 Bedingungen (W-Z) mit jeweils 4 Parameters mit (M und SD). Hinweiß: immer die selbe anzahl an Nachkommastellen angeben.

Algorithmus A.1 Sample algorithm

```
procedure SAMPLE($a$,$v_e$)
    $\mathsf{parentHandled} \leftarrow (a = \mathsf{process}) \vee \mathsf{visited}(a'), (a', c, a) \in \mathsf{HR}$
                                        // $(a', c'a) \in \mathsf{HR}$ denotes that $a'$ is the parent of $a$
    if $\mathsf{parentHandled} \wedge (\mathcal{L}_{in}(a) = \emptyset \vee \forall l \in \mathcal{L}_{in}(a) : \mathsf{visited}(l))$ then
        $\mathsf{visited}(a) \leftarrow$ true
        $\mathsf{writes}_\circ(a, v_e) \leftarrow \begin{cases} \mathsf{joinLinks}(a, v_e) & |\mathcal{L}_{in}(a)| > 0 \\ \mathsf{writes}_\circ(p, v_e) & \exists p : (p, c, a) \in \mathsf{HR} \\ (\emptyset, \emptyset, \emptyset, false) & \text{otherwise} \end{cases}$
        if $a \in \mathcal{A}_{basic}$ then
            HANDLEBASICACTIVITY($a$,$v_e$)
        else if $a \in \mathcal{A}_{flow}$ then
            HANDLEFLOW($a$,$v_e$)
        else if $a = \mathsf{process}$ then                // Directly handle the contained activity
            HANDLEACTIVITY($a'$,$v_e$), $(a, \perp, a') \in \mathsf{HR}$
            $\mathsf{writes}_\bullet(a) \leftarrow \mathsf{writes}_\bullet(a')$
        end if
        for all $l \in \mathcal{L}_{out}(a)$ do
            HANDLELINK($l$,$v_e$)
        end for
    end if
end procedure
```

Und wer einen Algorithmus schreiben möchte, der über mehrere Seiten geht, der kann das nur mit folgendem **üblen** Hack tun:

Algorithmus A.2 Description

```
code goes here
test2
```

A.8. Abkürzungen

Beim ersten Durchlaf betrug die Fehlerrate (FR) 5. Beim zweiten Durchlauf war die FR 3.

Mit `\ac{...}` können Abkürungen eingebaut werden, beim ersten aufrufen wird die lange Form eingesetzt. Beim wiederholten Verwenden von `\ac{...}` wird automatisch die kurz Form angezeigt. Außerdem wird die Abkürzung automatisch in die Abkürzungsliste eingefügt.

Definiert werden Abkürzungen in der Datei *ausarbeitung.tex* im Abschnitt '%%% acro' mithilfe von `\DeclareAcronym{...}{...}`.

Mehr infos unter: http://mirror.hmc.edu/ctan/macros/latex/contrib/acro/acro_en.pdf

A.9. Verweise

Für weit entfernte Abschnitte ist "varioref" zu empfehlen: "Siehe Appendix A.3 on page 18". Das Kommando `\vref` funktioniert ähnlich wie `\cref` mit dem Unterschied, dass zusätzlich ein Verweis auf die Seite hinzugefügt wird. `vref`: "Appendix A.1 on page 17", `cref`: "Appendix A.1", `ref`: "A.1".

Falls "varioref" Schwierigkeiten macht, dann kann man stattdessen "cref" verwenden. Dies erzeugt auch das Wort "Abschnitt" automatisch: Appendix A.3. Das geht auch für Abbildungen usw. Im Englischen bitte `\Cref{...}` (mit großen "C" am Anfang) verwenden.

A.10. Definitionen

Definition A.10.1 (Title)
Definition Text

Definition A.10.1 zeigt . . .

A.11. Verschiedenes

Kapitälchen werden schön gesperrt...

I. Man kann auch die Nummerierung dank paralist kompakt halten
II. und auf eine andere Nummerierung umstellen

A.12. Weitere Illustrationen

Abbildungen A.4 und A.5 zeigen zwei Choreographien, die den Sachverhalt weiter erläutern sollen. Die zweite Abbildung ist um 90 Grad gedreht, um das Paket `rotating` zu demonstrieren.

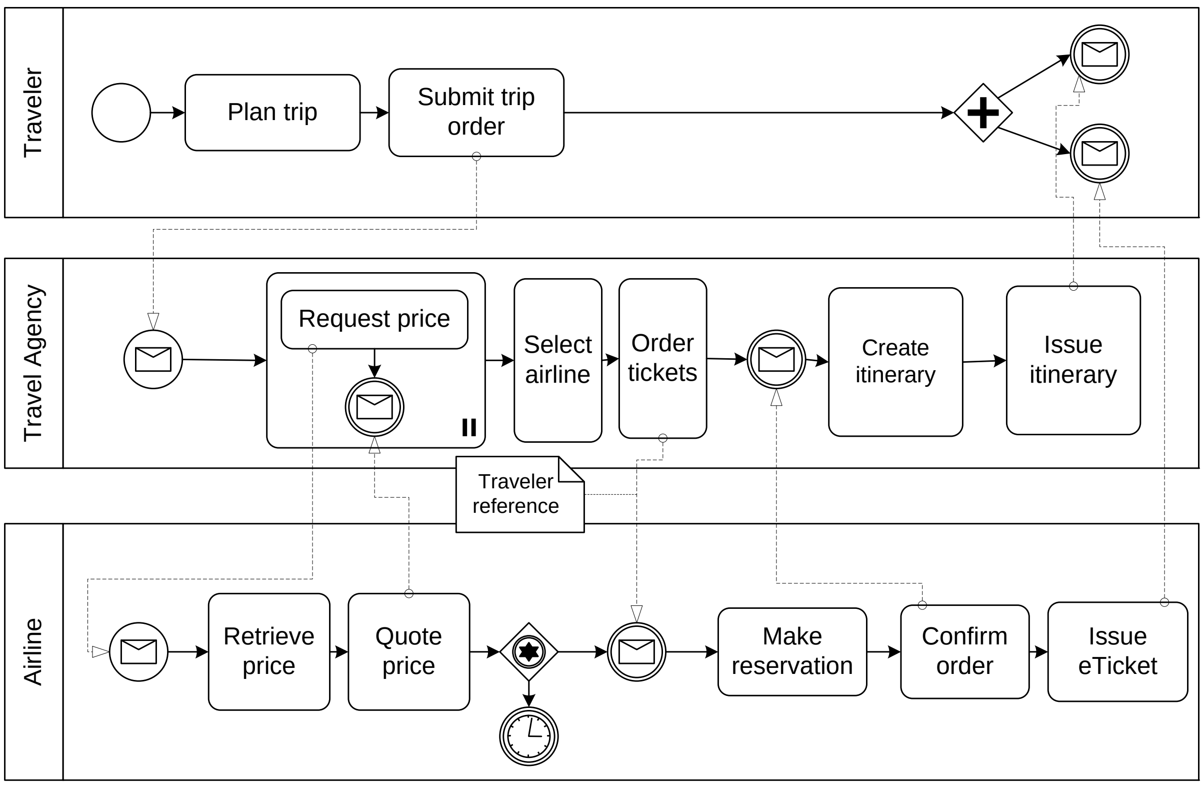

Figure A.4.: Beispiel-Choreographie I

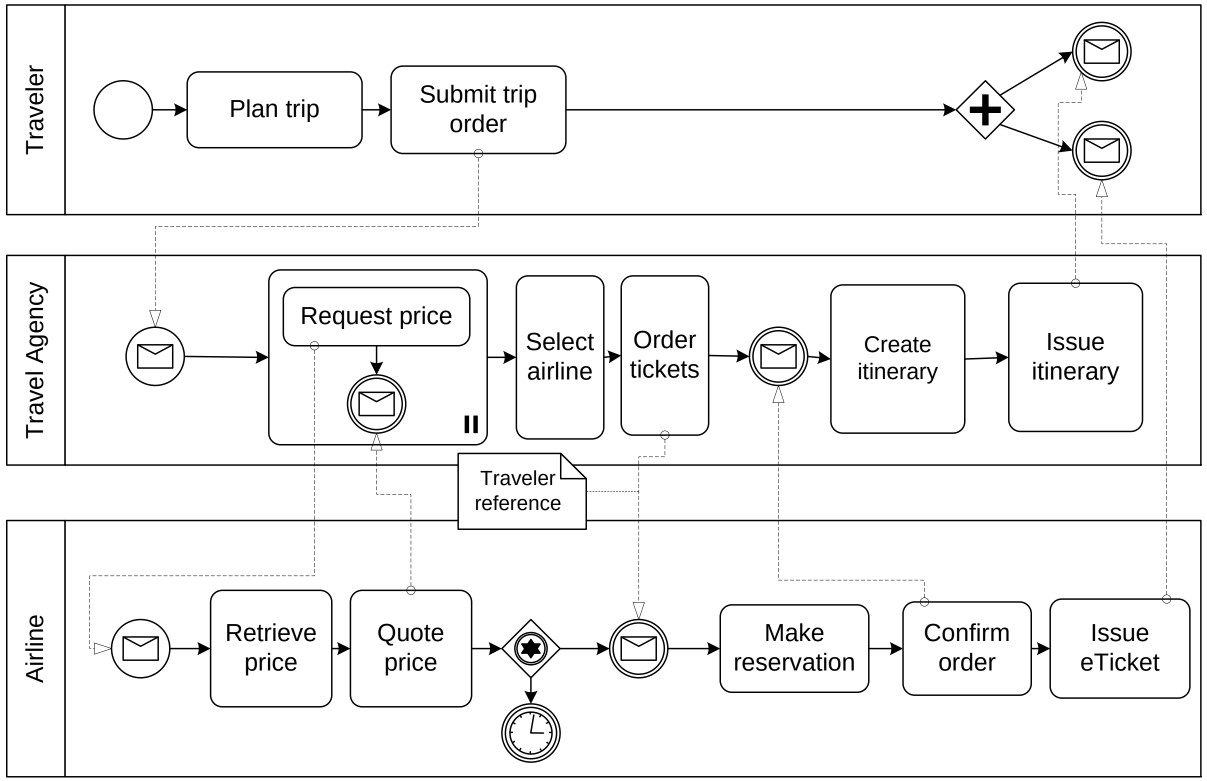

Figure A.5.: Beispiel-Choreographie II

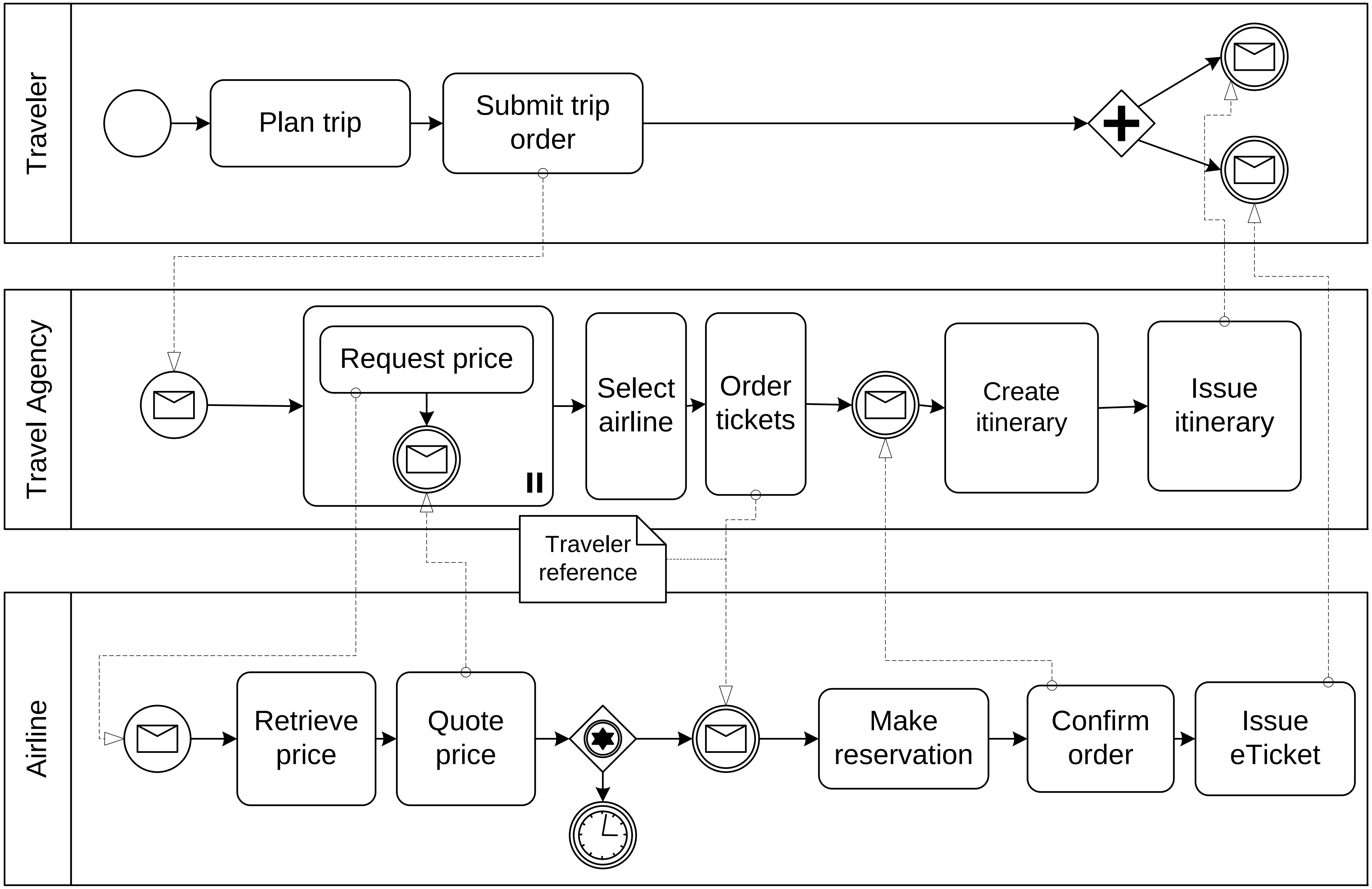

Figure A.6.: Beispiel-Choreographie, auf einer weißen Seite gezeigt wird und über die definierten Seitenränder heraus-ragt

A.13. Schlusswort

Verbesserungsvorschläge für diese Vorlage sind immer willkommen. Bitte bei github ein Ticket eintragen (https://github.com/latextemplates/uni-stuttgart-computer-science-template/issues).

All links were last followed on March 17, 2008.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature