Datenflussmodellierung für Vertraulichkeitsanalysen in Palladio

Proposal für die Bachelorarbeit von

Thomas Czogalik

an der Fakultät für Informatik
Institut für Programmstrukturen und Datenorganisation (IPD)

Erstgutachter:	Prof. Dr. Ralf H. Reussner
Zweitgutachter:	Jun.-Prof. Dr.-Ing. Anne Koziolek
Betreuender Mitarbeiter:	M. Sc. Stephan Seifermann
Zweiter betreuender Mitarbeiter:	Dipl.-Inform. Emre Taşpolatoğlu

01. Juni 2016 – 31. September 2016

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Inhaltsverzeichnis

1 Einleitung

Heutige IT-Systeme verarbeiten immer mehr vertrauliche Daten und werden gleichzeitig komplexer. Die Vertraulichkeit der Daten soll innerhalb der Komponenten, auf dem jeweiligen Gerät sowie beim Übertragen zwischen Software-Komponenten oder Geräten gewährleistet werden. Dies ist eine große Herausforderung. Außerdem ist eine Bewertung, ob das System ausreichend abgesichert ist schwierig, da in komplexen Systemen unklar ist, wo relevante Daten verarbeitet werden.

Dies ist mithilfe von Datenflussanalysen möglich. Diese Analysen nutzen die Implementierung der Anwendung als Eingabe. Fehler können dadurch aber erst nach der Implementierung erkannt und nicht bereits vorzeitig beseitigt werden. Ist ein Fehler durch eine mangelhafte Architektur entstanden, muss ggf. sogar ein Großteil der Implementierung verworfen werden. Ansätze zum Erkennen solcher Fehler auf Architekturebene sind in Entwicklung, aber scheitern daran, dass Datenflüsse auf Architekturebene bisher nur indirekt, zum Beispiel über Parameterübergabe spezifizierbar sind.

Ziel dieser Bachelorarbeit ist die Modellierung von Datenflüssen auf Architekturebene zu ermöglichen. Diese Modellierung soll für eine Vertraulichkeitsanalyse in Palladio genutzt werden. Sie soll aber auch für andere auf Datenflüssen aufbauende Analysen, beispielsweise aus dem Sicherheitsbereich nutzbar sein.

Nachdem der Datenfluss modelliert wurde, soll es möglich sein sicherheitsrelevante Eigenschaften zu definieren. Auch der Standort der Hardware soll spezifizierbar sein. Dabei soll verhindert werden, dass die Vertraulichkeit der Daten gefährdet wird, wenn diese an einen Ort geraten, an dem Vertraulichkeit nicht garantiert werden kann.
Außerdem soll die Möglichkeit gegeben werden Verbindungen zwischen Komponenten oder Geräten zu charakterisieren, beispielsweise dadurch, ob auf der Verbindung die Daten verschlüsselt übertrage werden oder nicht.

Im folgenden Kapitel 2 werden die Grundlagen erläutert, die für die geplante Bachelorarbeit benötigt werden. In Kapitel 3 wird ein Überblick über verwandte Arbeiten gegeben. Die geplanten Tätigkeiten der Bachelorarbeit werden in Kapitel 4 beschrieben. Die Beschreibung der Validierung erfolgt in Kapitel 5. In den Kapiteln 6 und 7 werden die Organisation und der Zeitplan sowie Risikomanagement erläutert.

2 Grundlagen

Im folgenden Abschnitt werden grundlegende Konzepte und Begriffe erläutert, die für die geplante Bachelorarbeit benötigt werden.

2.1 Vertraulichkeit

In der Literatur gibt es verschiedene Definitionen für Vertraulichkeit. In der Europäischen-Datenschutzgrundverordnung (EU-DSGVO) heißt es, dass Vertraulichkeit die Eigenschaft ist, dass Unbefugte keinen Zugang zu den Daten haben und weder die Daten noch die Geräte, mit denen diese verarbeitet werden, benutzen können (vgl. Art. 26ff EU-DSGVO). Eine andere Definition liefert das Bundesverfassungsgericht (BVerfG) in einem Urteil. Laut diesem versteht man unter Vertraulichkeit den Schutz vorhandener Daten gegen das Ausspähen (vgl. BVerfG 120, 274). Eine weitere Definition findet sich in dem Buch **Secure Systems Development with UML** [2]. Dort heißt es, dass Daten nur von legitimen Parteien gelesen werden dürfen. Eine Festlegung auf einen Vertraulichkeitsbegriff soll hier noch nicht geschehen. Dies soll im Zuge der Spezifizierung der Vertraulichkeit (vgl. 4.1.1) als eigenständige Arbeit geschehen.

2.2 Modellgetriebene Software-Entwicklung

In der modellgetriebenen Software-Entwicklung [9] werden Teile des Software-Systems mit Hilfe von Modellen auf einem höheren Abstraktionslevel beschrieben. Dabei fließen die Modelle in die Software mit ein und sind Teil des Endproduktes. Das Ziel der modellgetriebenen Softwareentwicklung ist die Verbesserung der Softwarequalität und der Wiederverwendbarkeit. Außerdem soll eine Erhöhung der Entwicklungseffizienz zum Beispiel durch Quellcodeerzeugung aus den Modellen erreicht werden.

2.2.1 Modell

Nach Herbert Stachowiak [8] zeichnet sich ein Modell durch die folgenden drei Merkmale aus.

- **Abbildung** - Bei einem Modell handelt es sich um eine Abbildung oder Repräsentation von einem künstlichen oder natürlichen Original. Das Original kann selbst wieder ein Modell sein.
- **Verkürzung** - Es werden nicht alle Attribute des Originals erfasst, sondern nur diejenigen, die relevant sind.

- **Pragmatismus** - Modelle erfüllen ihre Ersatzfunktion für bestimmte Subjekte, innerhalb einer bestimmten Zeit und unter Einschränkung bestimmter gedanklicher oder tätlicher Operationen. Beispielsweise benötigt eine Analyse je nach gewünschter Genauigkeit auch unterschiedlich genaue Modelle. Der Pragmatismus schreibt vor, dass man hier nicht unnötig genau arbeitet.

Ein Original könnte zum Beispiel eine Audio-Datei sein und das dazugehörige Modell könnte ein Lied-Element mit Titel “Stairway to Haven“, des Sängers “Led Zeppelin“ und dem Jahr “1971“ sein.

2.2.2 Meta-Modelle

Ein Meta-Modell beschreibt die Struktur eines Modells. In dem Musik-Beispiel, würde das Meta-Modell eine Klasse **Song** mit den Attributen **Titel**, **Sänger**, vom Typ String und **Jahr** vom Typ Integer enthalten. Ein Meta-Modell muss folgende Bereiche abdecken:

- **Abstrakte Syntax**: Sie beschreibt die Konstrukte, aus denen die Modelle bestehen, sowie deren Eigenschaften und Beziehungen.
- **Konkrete Syntax**: Diese beschreibt die Darstellung der Konstrukte, Eigenschaften und Beziehungen, die in der abstrakten Syntax spezifiziert sind.
- **Statische Semantik**: Mithilfe dieser werden Modellierungsregeln und Einschränkungen ausgedrückt, die mit der abstrakten Syntax nicht möglich sind.
- **Dynamische Semantik**: Sie drückt die Bedeutung der Konstrukte aus und wird oft nicht formal, sondern durch natürlichsprachlichen Text spezifiziert.

2.3 Rollen der komponentenbasierten Entwicklung

Die Modellierung des Datenflusses erfolgt auf Software-Architektur-Modellen. Dabei gibt es vier verschiedene Rollen [3], die an unterschiedlichen Teilen der Architektur arbeiten und dort ihr spezifisches Wissen einbringen.

- **Systemarchitekt**: Die Architektur und der Zusammenhang der einzelnen Komponenten miteinander werden vom Systemarchitekten entworfen. Er ist auch dafür zuständig weitere Anweisungen an die anderen Rollen weiterzugeben.
- **Domänenxperte**: Das Wissen darüber, wie der Benutzer mit dem System interagiert und welche Parameter im Kontrollfluss verwendet werden, stammt vom Domänenxperten.
- **Komponentenentwickler**: Für das Implementieren und Spezifizieren der einzelnen Komponenten ist der Komponentenentwickler verantwortlich.
- **Softwareverteilungsexperte**: Die Zusammenstellung der Systemumgebung der Software wird vom Softwareverteilungsexperten übernommen. Außerdem weist der Softwareverteilungsexperte den Komponenten Ressourcen zu.

2.4 Palladio Komponentenmodell (PCM)

Das Palladio Komponentenmodell (PCM) [1] ist eine Architecture-Description-Language (ADL). Sie wird verwendet um komponentenbasierte Software zu beschreiben. Neben der Beschreibung sind auch Qualitätsanalysen, wie etwa bezüglich der Performanz oder Zuverlässigkeit möglich. Im Folgenden werden Teile des PCM erklärt, die für die Bachelor Arbeit wichtig sind.

2.4.1 Submodelle

Auch im PCM werden vier Rollen betrachtet. Diese entsprechen den Rollen aus 2.3. Der einzige Unterschied ist dabei, dass im PCM der Systemarchitekt Software-Architekt heißt. Die Aufgaben sind jedoch die selben.
Jede dieser Rollen ist außerdem für ein bestimmtes Submodell zuständig. Für das **Component-Repository-Modell** ist der Komponentenentwickler zuständig. Dieses Modell enthält die einzelnen Komponenten und Schnittstellen. Der Software-Architekt ist für das **System-Modell** zuständig, das die Zusammensetzung der Komponenten charakterisiert. Der Softwareverteilungsexperte ist für gleich zwei Modelle verantwortlich. Das erste Modell, ist das **Execution-Environmet-Modell**. Es beschreibt die benutzte Hardware und das Netzwerk. Das zweite Modell beschreibt wie die einzelnen Komponenten auf der Hardware verteilt werden. Dieses Modell ist das **Component-Allocation-Modell**. Schließlich ist der Domänenxperte für das **Usage-Modell** zuständig. Dieses beschreibt die Interaktion des Benutzers mit dem System.
Diese Modelle werden im Rahmen der Bachelorarbeit entsprechend um die Möglichkeit zur Modellierung von Datenflüsse und ggf. weiteren für die Vertraulichkeitsanalyse notwendigen Eigenschaften erweitert.

2.4.2 Service Effect Specification (SEFF)

Mithilfe der Service Effect Specification (SEFF) [7] kann das Verhalten innerhalb der Komponenten auf abstrakten, auf bestimmte Qualitätsattribute zugeschnittenen Leveln beschrieben werden. Durch das spezifizieren von SEFFs kann man die Software auf Architekturebene analysieren. Dabei gibt es drei verschiedene Abstraktionslevel, die aufeinander aufbauen:

- **Signature-List-Based-Interface**: Schnittstellen dienen zur Kommunikation zwischen den Komponenten. Auf dem Signature-List-Based-Interface bauen die anderen Abstraktionslevel auf. Es besteht aus Parametern und Rückgabewerten.
- **Protocol-Modeling-Interface**: Mithilfe dieser Schnittstelle lassen sich Aufrufsequenzen definieren.
- **Quality-Of-Service-Modelling Interface**: Hier können verschiedene Qualitätsattribute definiert werden.

Der RDSEFF ist eine Erweiterung der SEFF. Die Performanzvorhersage für Komponenten basiert in Palladio unter anderem auf der Notation des RDSEFFs.

Im Rahmen der Bachelorarbeit soll eine weitere SEFF-Spezialisierung entstehen, die den Datenfluss innerhalb der Komponenten beschreibt.

3 Verwandte Arbeiten

Im Folgenden werden Arbeiten vorgestellt, die ähnliche Aufgabenstellungen haben oder Teile der Bachelorarbeit behandeln.

Für die Spezifizierung der Vertraulichkeit in Abschnitt 4.1.1 werden die folgenden Arbeiten verglichen um die verschiedenen Aspekte von Vertraulichkeit zu beleuchten und notwendige Eingabedaten für Vertraulichkeitsanalysen zu ermitteln.

- In der erste Arbeit **Specification and Verification of Confidentiality in Component-Based Systems** [4] wird Vertraulichkeit spezifiziert indem Anforderungen aufgestellt werden, die von einem vertraulichen System erfüllt werden müssen.

- In dem Buch **Secure System Development for UML** [2] werden zunächst einige Sicherheitseigenschaften spezifiziert (unter anderem Vertraulichkeit) um eine Sicherheitsanalyse für UML-Diagramme durchführen zu können.

- In der Arbeit **A Security Confidential Document Model and Its Application** [11] wird ein Model vorgestellt, mit dem die Vertraulichkeit von elektronischen Dokumenten sichergestellt werden soll. Hier werden auch Anforderungen für Vertraulichkeit spezifiziert.

In dem Buch **Principles of Program Analysis** [6] wird die Datenflussanalye auf Implementierungsebene beschrieben. Obwohl in der Bachelorarbeit Datenflüsse auf Architekturebene betrachtet werden, kann das Buch als Grundlagenlektüre für Datenflussanalysen dienen.
In der Arbeit **Model-Driven Specification and Analysis of Confidentiality in Component-Based Systems** [5] werden Datenflüsse auf Architekturebene betrachtet, aber nicht innerhalb der Komponenten. Im Laufe der Bachelorarbeit soll die dort erarbeite Vertraulichkeitsanalyse für die Validierung (vgl. Kapitel 5) dienen.

Außerdem konnten drei Paladio-Erweiterungen identifiziert werden, die sich mit dem Problem der Datenflussmodellierung befassen.

- **Model-Driven Specification and Analysis of Confidentiality in Component-Based Systems** [1] ist eine Erweiterung, die die Vertraulichkeit von Informationen spezifiziert und analysiert, die in komponentenbasierten Systemen verarbeitet werden

[1] `https://sdqweb.ipd.kit.edu/wiki/Model-Driven_Specification_and_Analysis_of_Confidentiality_in_Component-Based_Systems`

- **PASE** [2] erlaubt in PCM-Modellen den Datenfluss zu modellieren und analysieren.
- **Palladio.TX** [3] modelliert und analysiert transaktionale Informationssysteme und spezifiziert dazu Daten, sowie deren Speicherung in Datenbanktabellen.

Die drei Erweiterungen liefern jedoch keine oder keine ausreichende Dokumentation. Nur **PASE** liefert auf seiner Wiki-Seite eine Beschreibung der Erweiterung des Metamodells. Außerdem kann keine der Erweiterungen feststellen, wie Daten innerhalb von Komponenten verarbeitet werden. Dies ist jedoch für eine Vertraulichkeitsanalyse notwendig.

[2] https://sdqweb.ipd.kit.edu/wiki/PASE
[3] https://sdqweb.ipd.kit.edu/wiki/Palladio.TX

4 Konzeption

Im folgendem Abschnitt werden die geplanten Tätigkeiten beschrieben.

4.1 Anforderungserhebung für die Vertraulichkeitsanalyse

4.1.1 Spezifiziere Vertraulichkeit

Da in verwandter Literatur keine einheitliche Spezifikation und Anforderungen an Vertraulichkeit zu finden sind, werden im ersten Schritt die einzelnen Ansätze verglichen und Überschneidungen ermittelt. Sollten diese Anforderungen für Vertraulichkeit relevant sein, werden diese für die spätere Modellierung berücksichtigt. Zum Beispiel findet sich in [2] und [4] die Anforderung, dass man auf Daten nur mit Erlaubnis zugreifen kann und diese einem zunächst erteilt werden muss.
Nachdem eine Sammlung von Anforderungen an Vertraulichkeit entstanden ist, werden die Anforderungen bezüglich ihrer Allgemeingültigkeit für Analysen priorisiert. Bei der Modellierung werden dann häufig genutzte Anforderungen zuerst umgesetzt.

4.1.2 Modellierung von Daten und Datenfluss

Nachdem die Vertraulichkeit spezifiziert wurde, werden im nächsten Schritt die verschiedenen Rollen der komponentenbasierten Entwicklung identifiziert. Daraufhin wird ermittelt, welche Rolle welche Informationen liefern kann. Im Anschluss werden die einzelnen Sichten identifiziert, die zu der jeweiligen Rolle gehören. Schließlich werden die identifizierten Daten auf die einzelnen Modelle abgebildet.

4.2 Modellierung in Palladio

4.2.1 Vorgehen bei der Implementierung

Schließlich soll die Datenflussmodellierung in Palladio implementiert werden. Dazu muss zunächst von den allgemeinen Rollen auf die Palladio Rollen abgebildet werden. Im nächsten Schritt müssen die Meta-Modelle der jeweiligen Sicht erweitert werden.
Die Implementierung besteht aus mehreren Teilen, die sich im Verlauf der Anforderungerhebung ergeben. Ein Teil wird voraussichtlich sein einen SEFF zu implementieren, der den Datenfluss innerhalb einer Komponente beschreibt. Ein weiterer Teil sind Annotationen die zum Beispiel an Server angehängt werden können um deren Standort zu beschreiben. Annotationen sind ebenfalls an Verbindungen zwischen Rechenknoten sinnvoll, um beispielsweise anzuzeigen ob auf dem Link verschlüsselt kommuniziert wird.

Weitere Teile müssen noch im Laufe der Spezifizierung und Modellierung identifiziert werden.

4.2.2 Erweiterung der Meta-Modelle

Die Erweiterung des PCMs erfolgt durch die Erweiterung seines Metamodells. Dazu wurde von Strittmatter et al. [10] ein Ansatz entwickelt, bei dem die zu modellierenden Informationen in Gruppen eingeteilt werden. Das Meta-Modell wird in Schichten eingeteilt, die jeweils eine Gruppe abbilden. Schichten referenzieren sich gegenseitig. Dies führt zu einer modularen, flexiblen und erweiterbaren Struktur, die die Koexistenz von verschiedenen Qualitätsattributen und -analysen ermöglicht. Das Meta-Modell ist in die folgenden vier Schichten unterteilt:

- Die **Paradigm**-Schicht ist die Basisschicht. Sie legt den Grundstein für die Modellierungssprache, indem sie die Struktur, aber keine Semantik bereitstellt.
- Die Semantik für die abstrakte Struktur der Sprache liefert die **Domain**-Schicht.
- In der **Quality**-Schicht werden Qualitätseigenschaften für bestimmte Bereiche hinzugefügt.
- Schließlich bietet die **Analysis**-Schicht Module für die Eingabe, Ausgabe und den internen Zustand. Außerdem gibt es Konfigurationsoptionen für Analysen.

In der Bachelorarbeit wird die **Quality**-Schicht erweitert, da dort Qualitätseigenschaften spezifiziert werden können. Sonst wird keine Schicht erweitert. Die **Analysis**-Schicht wird nicht erweitert, aber für weitere Arbeiten berücksichtigt.

Nachdem die vorherigen Schritte abgearbeitet wurden, werden nun die einzelnen Anforderungen für Palladio iterativ implementiert. Dazu muss zunächst überlegt werden, welcher Mechanismus zu Erweiterung der Meta-Modelle am sinnvollsten ist.

Dabei gibt es verschiedene Möglichkeiten das Meta-Modell zu erweitern. Eine Möglichkeit wäre das existierende Meta-Modell durch Annotationen oder Stereotypen zu erweitern. Das Problem dabei ist, dass flache und unstrukturierte Informationen entstehen. Eine andere Möglichkeit wäre die Erweiterung mithilfe eines neuen Entwicklungszweiges. Dabei entsteht aber das Problem das duplizierte Teile gepflegt werden müssen, wenn sie im ursprünglichen Zweig verändert wurden.

5 Validierung

Im Laufe der Bachelorarbeit wird ein Ansatz zu Modellierung von Datenflüssen auf Architekturebene entwickelt. Um den Nachweis über die Einsatzeignung der entwickelten Methodik aus Abschnitt 4 zu erbringen, soll geprüft werden ob die Eigenschaften nach Stachowiak (vgl. Kapitel 2.2.1) erfüllt sind. Die Eigenschaften **Abbildung** und **Verkürzung** sind gegeben, da es sich um eine Abbildung von Datenflüssen handelt und nur relevante Attributen erfasst werden. Schließlich muss die Eigenschaft **Pragmatismus** geprüft werden. Pragmatismus ist gegeben, wenn die Modelle für ihren späteren Einsatzzweck geeignet sind. Im Rahmen der Bachelorarbeit sind Analysen der Einsatzzweck. Deshalb muss geprüft werden ob das Meta-Modell auch diese Eigenschaft unterstützt. Dafür gibt es zwei Ansätzen.

Der erste Ansatz ist die Vertraulichkeitsanalyse aus [5]. Diese Analyse prüft die Vertraulichkeit in komponentenbasierten Systemen. Für die Analyse müssen Eingabe und Ausgabe eines Systems spezifiziert werden, sowie Zugriffsspezifikation für Hardware und Kommunikationsverbindungen. Mithilfe dieser Informationen und einem Angreifer Modell wird eine Architektur- und Code-Analyse durchgeführt. Nach Durchführung der Analyse werden Designfehler, Datenlecks und Verstöße gegen die Spezifikation angezeigt. Die Analyse soll auf einem oder beiden Fallbeispielen aus der Arbeit laufen. Dabei handelt es sich um ein verteiltes Reiseplaner-System und um ein Cloud-Hosting-System mit zwei Tiers. Beide Systeme müssen vor der Analyse mit Datenflüssen ausgestattet werden. Dies geschieht, indem spezifiziert wird, welche Parameter sich innerhalb der Komponente beeinflussen. Für die Validierung muss die Analyse ebenfalls angepasst werden, damit der Datenfluss berücksichtigt wird.

Eine weitere Möglichkeit der Validierung kann mithilfe einer anderen Analyse durchgeführt werden. Die Analyse soll über Modellabfragen realisiert werden. Dazu kann zum Beispiel eine Analyse aus der Literatur genommen werden. Als Fallbeispiel soll der Palladio MediaStore [7] dienen, für den davor Datenflüsse modelliert werden.

6 Organisatorisches

Im folgenden Abschnitt wird beschrieben, wie die geplante Arbeit ablaufen soll.

6.1 Betreuer

Erstgutachter: Prof. Dr. Ralf H. Reussner
Zweitgutachter: Jun.-Prof. Dr.-Ing. Anne Koziolek
Erster Betreuer: M. Sc. Stephan Seifermann
Zweiter Betreuer: Dipl.-Inform. Emre Taşpolatoğlu

6.2 Artefakte

Im Laufe der Bachelorarbeit werden die folgenden Artefakte erzeugt werden:

- Anforderungen an Modellierungsumfang aus Vertraulichkeitsdefinitionen
- Erweiterungskonzept für Meta-Modelle der einzelnen Sichten
- Erweiterungskonzept der Meta-Modelel für Palladio
- Datenflussmodellierung für Palladio mithilfe einer Meta-Modell-Erweiterung
- Ausarbeitung der Bachelorarbeit
- Vortrag zur Bachelorarbeit

6.3 Entwicklungsumgebung und Werkzeuge

Die folgenden Entwicklungsumgebungen und Werkzeuge werden für die Bachelorarbeit benötigt:

- TexMaker
- Eclipse Mars
- Palladio 4.0
- MS Visio

7 Zeitplan und Risikomanagement

In diesem Abschnitt werden das Risikomanagement und der Zeitplan der Bachelorarbeit betrachtet.

7.1 Risikomanagement

Während der Bachelorarbeit müssen mehrere Risiken beachtet werden.

Es könnte passieren, dass in der Literatur keine Spezifizierung von Vertraulichkeit sowie deren Anforderung gefunden werden kann. Die Voruntersuchung hat jedoch gezeigt, dass dies unwahrscheinlich ist (vgl. Kapitel 3).

Wahrscheinlicher ist, dass die Modellierung, Implementierung, Validierung sowie das Schreiben der Bachelorarbeit, mehr Zeit in Anspruch nehmen als geplant. Um dem entgegen zu wirken sollten bei den regelmäßigen Treffen mit den Betreuern die Einhaltung des Zeitplans kontrolliert werden. Dadurch lassen sich Probleme frühzeitig erkennen, Iterationen weglassen und ggf. der Fokus neu setzen lassen.

Schließlich könnte der Fall auftreten, dass die geplante Validierung mit Hilfe der Vertraulichkeitsanalyse aus der Arbeit **Model-Driven Specification and Analysis of Confidentiality in Component-Based Systems** [5] zu aufwendig sein wird, da die Anpassung an die Datenflussmodellierung aus dieser Bachelorarbeit ggf. zu zeitaufwendig ist. Sollte dies der Fall sein, wird mit den Betreuern Rücksprache gehalten und auf die zweite Validierungsmethode zurückgegriffen, die einen geringeren Zeitumfang benötigen wird.

7.2 Zeitplan

Die Abbildung 7.2 zeigt den geplanten Ablauf der Bachelorarbeit. Damit der kontinuierliche Fortschritt der Bachelorarbeit gesichert ist, finden wöchentliche Treffen mit den Betreuern statt.

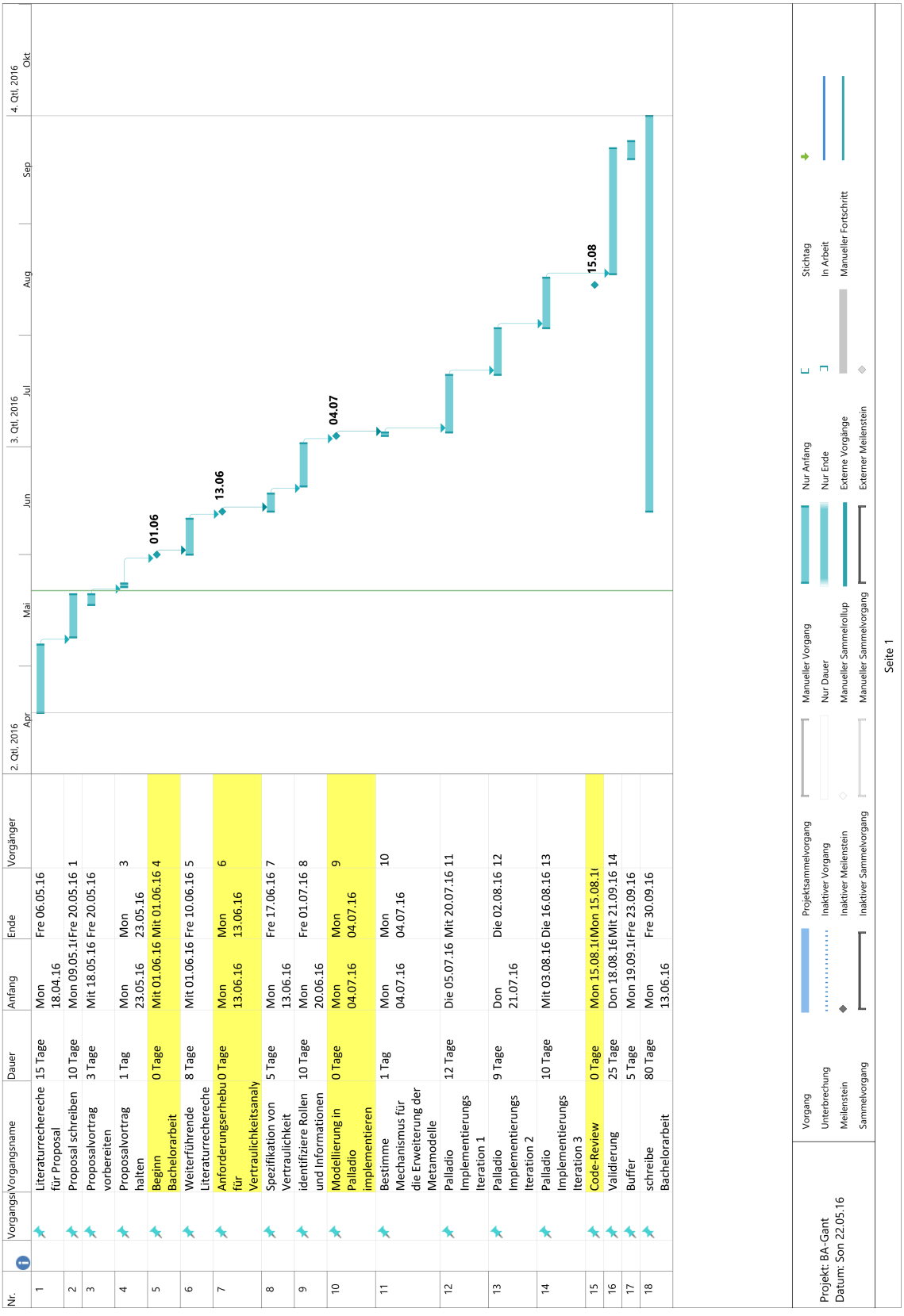

Abbildung 7.1: Zeitplan der Bachelorarbeit

Literatur

[1] Steffen Becker. "The palladio component model". In: *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering - WOSP/SIPEW '10* March (2010), S. 257. DOI: 10.1145/1712605.1712651. URL: http://portal.acm.org/citation.cfm?doid=1712605.1712651.

[2] Jan Jürjens. *Secure Systems Development with UML*. 2005.

[3] Heiko Koziolek und Jens Happe. "A QoS Driven Development Process Model for Component-Based Software Systems". In: *Proceedings of the 9th International Symposium on Component Based Software Engineering (CBSE 2006)* (2006), S. 336–343. DOI: 10.1007/11783565_25. URL: http://dx.doi.org/10.1007/11783565%7B%5C_%7D25.

[4] Max E Kramer u. a. *Poster: Specification and Verification of Confidentiality in Component-Based Systems Max*.

[5] Max E. Kramer u. a. *Specification and Verification of Confidentiality in Component-Based Systems*. Poster at the 35th IEEE Symposium on Security and Privacy. San Jose, California, USA, 2014. URL: http://www.ieee-security.org/TC/SP2014/posters/KRAME.pdf.

[6] Flemming Nielson, Hanne R. Nielson und Chris Hankin. *Principles of Program Analysis*. 1999.

[7] Ralf Reussner u. a. *Modeling and Simulating Software Architectures - The Palladio Approach*.

[8] Herbert Stachowiak. *Allgemeine Modelltheorie*. 1973, S. 131–133.

[9] Thomas Stahl u. a. *Modellgetriebene Softwareentwicklung Techniken, Engineering, Management*. 2007.

[10] Misha Strittmatter u. a. "A Modular Reference Structure for Component-based Architecture Description Languages". In: *2nd International Workshop on Model-Driven Engineering for Component-Based Systems (ModComp)*. CEUR, 2015, S. 36–41. URL: http://ceur-ws.org/Vol-1463/paper6.pdf.

[11] Shi Yuan Zheng und Jun Liu. "A secure confidential document model and its application". In: *Proceedings - 2010 2nd International Conference on Multimedia Information Networking and Security, MINES 2010*. 2010, S. 516–519. ISBN: 9780769542584. DOI: 10.1109/MINES.2010.115.