Evaluierung

6

Die in dieser Arbeit entwickelte modellbasierte Entwicklungsmethode wurde mittels zweier verschiedener Techniken evaluiert. Zum einen fand die Evaluierung des Einflusses der modellbasierten Entwicklungsmethode auf die SoC-Entwicklung, wie nachfolgend in Abschnitt 6.1 beschrieben, durch Interviews statt. Zum anderen wurde der als Ergebnis der modellgetriebenen Automatisierung erhaltene Code auf Korrektheit und Performance-Eigenschaften mittels Co-Simulation getestet und mit dem manuell implementierten Code verglichen. Das Vorgehen sowie das Ergebnis dieser Tests wird in Abschnitt 6.2 vorgestellt.

6.1 Interviewevaluierung

Der Einfluss der in dieser Arbeit entwickelten Methode auf die SoC-Entwicklung erfolgte mittels einer zweistufigen Interviewevaluierung. Dabei orientiert sich das hier gezeigte Vorgehen an der Methode für die qualitative Inhaltsanalyse nach Mayring [93]. Da jedoch die Evaluierung der in dieser Arbeit entwickelten modellbasierten Entwicklungsmethode im Mittelpunkt der Befragung steht, weicht das Vorgehen in einigen Punkten vom Standard ab. So wurde für die Interviewevaluierung ein strengerer Rahmen in Form eines Leitfadens gewählt, um den Fokus der Befragung auf das entsprechende für diese Arbeit relevante Themengebiet zu lenken [17].

Im diesem Kapitel wird zu Beginn in Abschnitt 6.1.1 das Vorgehen für die Datenerhebung beschrieben. Auf die Technik bei der Analyse der erhobenen Daten wird in Abschnitt 6.1.2 näher eingegangen. Die Beschreibung der Auswahl der Interview-Teilnehmer/-innen erfolgt in Abschnitt 6.1.3. Am Ende des Kapitels folgen in Abschnitt 6.1.4 die Vorstellung sowie eine Diskussion der Ergebnisse aus der Interviewevaluierung.

A. Kirchner, *Entwicklung von Methoden zur abstrakten Modellierung von Automotive Systems-on-Chips*, https://doi.org/10.1007/978-3-658-38437-1_6

6.1.1 Datenerhebung

Für die Datenerhebung wurden Interview-Teilnehmer/-innen der Firma *Robert Bosch GmbH* in Reutlingen aus dem Bereich der SoC-Entwicklung mittels zweier unterschiedlicher, jedoch aufeinander aufbauender Interviewmethoden befragt. Dabei wurde folgendes Vorgehen für die Datenerhebung verwendet (Abbildung 6.1 veranschaulicht ergänzend das Vorgehen der Datenerhebung):

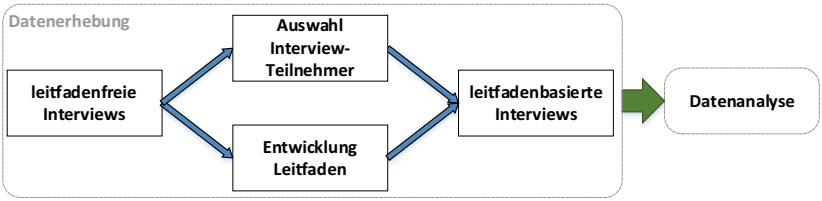

Abbildung 6.1 Datenerhebung Interviewevaluierung

- Im ersten Schritt wurden leitfadenfreie Interviews mit Entwicklern/-innen verschiedener Fachbereiche aus der SoC-Entwicklung über aus ihrer Sicht aktuell bestehende Defizite in der SoC-Entwicklung durchgeführt. Dazu wurde ein Fragenkatalog für die Befragung der Teilnehmer/-innen definiert, ohne dabei jedoch Antwortmöglichkeiten als Leitfaden vorzugeben. Die Teilnehmer/-innen sollten frei über die aus ihrer Sicht wichtigsten Defizite in der SoC-Entwicklung berichten.
- Darauf aufbauend wurden aus den leitfadenfreien Interviews fünf Defizite ausgewählt, welche von mehreren Teilnehmern/-innen der leitfadenfreien Interviews genannt wurden und sich aus deren Sicht stark auf die Effizienz der SoC-Entwicklung auswirken.
- Auf Grundlage dieser fünf ausgewählten Defizite und den erhaltenen Informationen aus den leitfadenfreien Interviews wurde ein Leitfaden für die leitfadenbasierten Interviews und damit die Interviewevaluierung erarbeitet.
- In der zweiten Stufe der Interviewevaluierung wurden Teilnehmer/-innen ausgewählt, welche bereits mehrjährige Erfahrungen in der SoC-Entwicklung vorweisen sowie bereits an der Entwicklung eines Pilotprojekts nach der neuen modellbasierten Entwicklungsmethode beteiligt waren.

- Bei diesen Interviews wurden den Teilnehmer/-innen im Leitfaden definierte Fragen unterschiedlicher Themenschwerpunkte gestellt, bei deren Beantwortung zudem eine vordefinierte Anzahl an Antworten zu Verfügung stand. Die Fragen sowie die Antwortwortmöglichkeiten der leitfadenbasierten Interviews werden in Abschnitt 6.1.4 näher behandelt.

Auf der Grundlage der erhobenen Daten erfolgte deren Analyse und Quantifizierung, wie im nachfolgenden Abschnitt beschrieben.

6.1.2 Datenanalyse der leitfadenbasierten Interviews

Die Fragenkomplexe 1–3 sollten dazu dienen, auf Grundlage der Einschätzungen bzw. Antworten der Interview-Teilnehmer/-innen Defizite in der SoC-Entwicklung, Ursachen für die Defizite und mögliche Lösungen zur Behebung der Ursachen zu bewerten. Dazu wurden die Teilnehmer/-innen für die Fragenkomplexe 1–3 jeweils um eine Priorisierung der fünf Antwortmöglichkeiten in fünf Priorisierungsstufen gebeten.

Die Definition der Priorisierungsstufen erfolgte dabei nach dem Vorbild einer Ratingskala. Dabei wird zwischen einer bipolaren Ratingskala, welche zwei gegensätzliche Dimensionen wie beispielsweise *sehr schlecht* bis *sehr gut* abbildet, und einer unipolaren Ratingskala, auf der lediglich eine Dimension abgebildet wird, unterschieden. Aufgrund der Form der in dieser Arbeit definierten Fragestellungen und Antwortmöglichkeiten für die Fragenkomplexe 1–3 wurde eine Definition der Priorisierungsstufen vergleichbar einer unipolaren Ratingskala gewählt. Das hier gewählte Vorgehen weicht jedoch maßgeblich von einer üblichen Befragung mittels Ratingskala ab. So sollen die Interview-Teilnehmer/-innen nicht, wie es in einer Befragung mittels Ratingskala üblich ist, für jede Fragestellung einen Wert der Ratingskala wählen, sondern die Antwortmöglichkeiten mittels Priorisierung in einer Art Ranking (Rangliste) darstellen. Dabei sollten die Priorisierungsstufen zum einen das Maß der Zustimmung seitens der Interview-Teilnehmer/-innen zur jeweiligen Antwortmöglichkeit im Ranking abbilden, zum anderen wird die Zustimmung, wie nachfolgend gezeigt, in Form einer prozentualen Gewichtung repräsentiert [94].

Priorisierungsstufe 1: (prozentuale Gewichtung: 100 %)
Priorisierungsstufe 1 bedeutet eine Gewichtung der Zustimmung mit 100 % für die Mittelwertberechnung und kann somit als **höchste** bzw. **volle Zustimmung** interpretiert werden.

Priorisierungsstufe 2: (prozentuale Gewichtung: 75 %)
Priorisierungsstufe 2 wird mit einer prozentualen Gewichtung von 75 % in der Mittelwertbildung verrechnet und ist als **hohe Zustimmung** zu interpretieren.
Priorisierungsstufe 3: (prozentuale Gewichtung: 50 %)
Priorisierungsstufe 3 lässt sich als **mittlere Zustimmung** interpretieren und erhält daher die prozentuale Gewichtung von 50 % für die Mittelwertberechnung.
Priorisierungsstufe 4: (prozentuale Gewichtung: 25 %)
Wählt einer/eine der Interview-Teilnehmer/-innen für eine Antwortmöglichkeit die Priorisierungsstufe 4, entspricht dies einer prozentualen Gewichtung für die Mittelwertberechnung von 25 % und lässt sich als **geringe Zustimmung** interpretieren.
Priorisierungsstufe 5: (prozentuale Gewichtung: 0 %)
Für den Fall, dass einer/eine der Interview-Teilnehmer/-innen einer Antwortmöglichkeit **keine Zustimmung** gibt, wird diese als Priorisierungsstufe 5 und damit mit 0 % prozentualer Gewichtung verrechnet.

Durch die Zuordnung von Priorisierungsstufen und prozentualer Gewichtung wurde die Quantifizierung der Zustimmung erreicht. Mithilfe der so erhaltenen prozentualen Werte konnte für jede Antwortmöglichkeit der Fragenkomplexe 1–3, wie in Abbildung 6.2 veranschaulicht, ein Mittelwert über alle Interview-Teilnehmer/-innen hinweg gebildet werden. Der so erhaltene Mittelwert, welcher aus den gewichteten Einzelwerten der Interview-Teilnehmer/-innen berechnet wurde, wird nachfolgend als gewichtete Zustimmungsrate (gZR) bezeichnet.

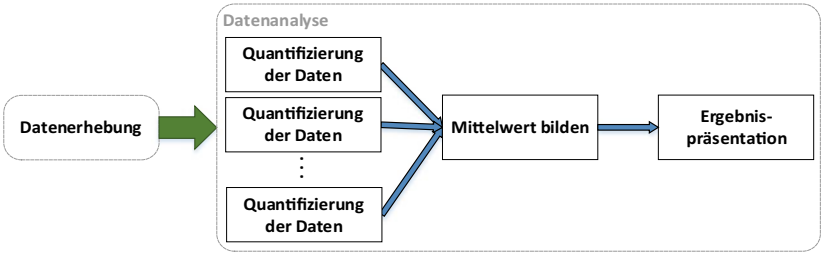

Abbildung 6.2 Datenanalyse Interviewevaluierung

Durch die Berechnung der gewichteten Zustimmungsrate für die jeweilige Antwortmöglichkeit konnte eine repräsentative Priorisierung der Antwortmöglichkeiten aller Teilnehmer/-innen erreicht werden.

Verglichen mit den Fragenkomplexen 1–3 wurde ein separates Vorgehen für die Datenanalyse im Fragenkomplex 4 gewählt. Dieser Unterschied beim Vorgehen der Datenanalyse für den Fragenkomplex 4 resultiert aus einer grundsätzlich anderen Fragestellung. Ziel des Fragenkomplexes 4 ist die Analyse des durch die Interview-Teilnehmer/-innen geschätzten Einflusses der modellbasierten Entwicklungsmethode auf den Aufwand verschiedener Bereiche der SoC-Entwicklung. Daher wurde für den Fragenkomplex 4 eine Befragung mittels bipolarer Ratingskala gewählt. Das bedeutet, den Interview-Teilnehmern/-innen wurden die sieben nachfolgend gezeigten Antwortmöglichkeiten zur Auswahl gestellt. Jede Antwort entspricht dabei einer Aussage über den erwarteten Einfluss der modellbasierten Entwicklungsmethode auf den Aufwand verschiedener Bereiche der SoC-Entwicklung aus Sicht der Interview-Teilnehmer/-innen. Die Interview-Teilnehmer/-innen wurden somit gebeten, eine der Antwortmöglichkeiten auszuwählen. Zusätzlich wurde auch hier eine prozentuale Quantifizierung der Skalenwerte vorgenommen. Die hier definierte Ratingskala teilt sich wie folgt auf:

- stark verringert (> 50 %)
- mittel bis stark verringert (25–50 %)
- leicht verringert (< 25 %)
- unverändert (0 %)
- leicht erhöht (< 25 %)
- mittel bis stark erhöht (25–50 %)
- stark erhöht (> 50 %)

Eine Verringerung bzw. Erhöhung des Aufwandes durch die Anwendung der neuen Methode wurde im Vorfeld auf Basis der leitfadenfreien Interviews auf den Bereich 0 %–50 % geschätzt. Daher wurde die Aufteilung des Wertebereichs mit einer feineren Aufteilung des Bereichs unter 50 % gelegt. Die Zustimmungsrate (ZR) ergibt sich für den Fragenkomplex 4 aus dem prozentualen Anteil der Interview-Teilnehmer/-innen, welche der jeweiligen Antwort und damit der Abschätzung zugestimmt haben. Die Ergebnispräsentation sowie die Diskussion aller Interview-Ergebnisse folgt in Abschnitt 6.1.4.

6.1.3 Interview-Teilnehmer/-innen der leitfadenbasierten Interviews

Im Zuge der in dieser Arbeit durchgeführten Interviewevaluierung wurden zehn Interview-Teilnehmer/-innen der Firma *Robert Bosch GmbH* in Reutlingen aus dem Bereich der SoC-Entwicklung mit verschiedenen Schwerpunkten ausgewählt. Dabei wurde darauf geachtet, dass die Interview-Teilnehmer/-innen bereits mehrere Jahre Erfahrung bei der Anwendung der aktuell bestehenden Entwicklungsmethode aufweisen und zudem bei der Anwendung der neuen modellbasierten Entwicklungsmethode auf ein Pilotprojekt beteiligt waren. Nachfolgend werden in Tabelle 6.1 die zehn Interview-Teilnehmer/-innen auf ihre jeweilige Tätigkeitsfelder aufgeteilt sowie jedem Teilnehmer und jeder Teilnehmerin für die spätere Diskussion der Ergebnisse und Aussagen eine Kennung zugeordnet.

Tabelle 6.1 Interview-Teilnehmer/-innen

Anzahl	Rolle	Kennung
3	Systemarchitekt/-innen	*SA1*, *SA2*, *SA3*
1	Verifikationsingenieur	*VI1*
1	Safety-Manager	*SM1*
4	SoC-Entwickler/-innen	*SE1*, *SE2*, *SE3*, *SE4*
1	Management	*MA1*

Wie bereits beschrieben, wurde bei der Auswahl der Teilnehmer/-innen darauf geachtet, dass die Befragten erste Erfahrungen mit der neuen modellbasierten Entwicklungsmethode erlangt hatten. Darüber hinaus wurde versucht, Entwickler/-innen verschiedener Tätigkeitsfelder auszuwählen, um so unterschiedliche Sichtweisen aus dem jeweiligen Tätigkeitsfeld in Hinblick auf die Evaluierung zu erhalten.

6.1.4 Ergebnisse und Diskussion

Wie bereits zu Beginn des Kapitels beschrieben, wurde vor den eigentlichen Evaluierungsinterviews leitfadenfreie Interviews durchgeführt, um bestehende Defizite in der SoC-Entwicklung sowie deren mögliche Ursachen zu erfragen. Auf Grundlage dieser Interviews wurde anschließend ein Leitfaden für die leifadenbasierte Interviewevaluierung extrahiert. Nachfolgend werden die jeweiligen

Evaluierungsfragen, die Antwortmöglichkeiten sowie die erhaltenen Ergebnisse der Interviewevaluierung vorgestellt. Dabei gibt es vier Abschnitte, welche jeweils einen der vier Fragenkomplexe behandeln. Im ersten Abschnitt werden die Befragung und Analyse der Defizite in der SoC-Entwicklung behandelt. Darauf aufbauend werden im zweiten Abschnitt die Ursachen für das jeweilige Defizit diskutiert. Im nächsten Abschnitt folgen die Beschreibung und die Diskussion der erhaltenen Ergebnisse zu den Lösungsansätzen des jeweiligen Defizits. Im letzten Abschnitt wird der Einfluss der modellbasierten Entwicklungsmethode auf den Aufwand in verschiedenen Bereichen sowie der gesamten SoC-Entwicklung behandelt.

Dadurch sollte die nötige Transparenz für die Beantwortung der Fragen geschaffen und den Interview-Teilnehmern/-innen zudem bei der Abgabe einer Antwort geholfen werden. Besonders bei der Beantwortung der Fragen aus dem Fragenkomplex 4, erleichterte die Zuordnung von prozentualen Werten zu den Antwortmöglichkeiten der Ratingskala die Abschätzung des Einflusses. Denn bei diesen Fragen wurden die Interview-Teilnehmer/-innen aufgefordert, eine Abschätzung über den Einfluss der modellbasierten Entwicklungsmethode auf den Aufwand bezogen auf die Teilbereiche und den gesamten SoC-Entwicklungs-Flow abzugeben.

Ergebnis und Diskussion: Defizite in der SoC-Entwicklung

Frage 1 befasst sich mit der Priorisierung der Defizite in Hinblick auf die negative Auswirkung des jeweiligen Defizites auf die Effizienz der SoC-Entwicklung. Dabei wurden aus dem Leitfaden fünf mögliche Antworten zur Verfügung gestellt, um deren Priorisierung die Interview-Teilnehmer/-innen von „höchste Zustimmung“ und damit „höchster negativer Einfluss“ nach „keine Zustimmung“ bzw. „kein negativer Einfluss“, wie in Abschnitt 6.1.1 und Abschnitt 6.1.2 beschrieben, gebeten wurden. Tabelle 6.2 zeigt die genaue Fragestellung, die zur Verfügung stehenden Antwortmöglichkeiten sowie die als Ergebnis der Befragung und anschließenden Analyse erhaltene gZR der Interviewevaluierung.

D1 (Implementierungsfehler) und *D2* (Verifikationsfehler) fokussieren sich auf die direkten Auswirkungen einer defizitären Spezifikation, auf die Implementierung bzw. die Verifikation und damit auf die Auswirkung des Defizites auf die Effizienz aufgrund der benötigten Fehlerbehebung.

Bei der Interviewevaluierung wurden Implementierungsfehler aufgrund einer unzureichenden Spezifikation mit 23 % gZR und Verifikationsfehler aufgrund einer unzureichenden Spezifikation mit 20 % gZR am niedrigsten priorisiert. Die Befragten *SE1*, *SE2*, *SM1* und *SE3* gaben zusätzlich an, dass aus ihrer Sicht entstandene Fehler in der Implementierung durch die Verifikation im aktuell bestehenden

Tabelle 6.2 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 1

Fragestellung	Antwortmöglichkeiten	gZR	Kennung
Welches der hier gezeigten Defizite hat den stärksten negativen Einfluss auf die Effizienz der SoC-Entwicklung? Bitte priorisieren Sie nach Stärke des Einflusses.	Implementierungsfehler aufgrund unzureichender Spezifikation	23 %	*D1*
	Verifikationsfehler aufgrund unzureichender Spezifikation	20 %	*D2*
	Nachträgliche Änderungen an Lasten- bzw. Pflichtenheft	80 %	*D3*
	Inkonsistenzen zwischen Spezifikationsdokumenten sowie zwischen Spezifikation und Entwurf	65 %	*D4*
	Fehlkommunikation zwischen Kunde und Entwickler sowie zwischen den Entwicklern/-innen	63 %	*D5*

Entwicklungs-Flow erkannt und behoben werden. Zudem wird bereits von Anfang an eine Behebung der Fehler in der Planung berücksichtigt.

D3 steht für den negativen Einfluss auf die Effizienz durch nachträgliche Änderungen an den vereinbarten Anforderungen zwischen Kunde und Entwickler. Solche nachträglichen Änderungen werden meist erst spät erkannt und deren Durchführung bindet in der Regel eine hohe Menge Ressourcen, führt zu einer Verschiebung des Zeitplans und kann laut *SE1* und *SE2* zu Folgefehlern führen. Die Folgefehler entstehen dabei aufgrund von nicht beachteten Einflüssen auf das umgebende System, welche durch die nachträglichen Änderungen hervorgerufen werden. Aufgrund der Häufigkeit, mit der das Defizit *D3* in der SoC-Entwicklung auftritt, und die schwerwiegenden Auswirkungen auf die Effizienz wurde das Defizit *D3* mit 80 % gZR am höchsten priorisiert.

Dem folgen *D4* (Inkonsistenzen Spezifikation/Entwurf) mit 65 % gZR und *D5* (Fehlkommunikation Entwickler-Kunde) mit 63 % gZR. Dabei wurde *D4* von den Interview-Teilnehmern/-innen *SA2*, *SM1* und *SE3* am höchsten priorisiert. Die stellenweise hohe Priorisierung resultiert laut *SE3* aus dem häufigen Auftreten des Defizits und der Gefahr, dass durch die Inkonsistenz Fehler im Entwurf durch die Verifikation nicht erkannt werden könnten.

D5 erhielt von fast allen Interview-Teilnehmern/-innen eine Priorisierung zwischen Priorisierungsstufe 1 und Priorisierungsstufe 3. Zudem wurde es besonders von den Systemarchitekten/-innen und dem Interview-Teilnehmer aus dem Managementbereich hoch priorisiert. Als Grund wurden dabei von *MA1* auch hier die

schwerwiegenden Folgen aus solchen Fehlkommunikationen für ein Entwicklungsprojekt genannt.

Aus den Interviews ging hervor, dass alle der hier behandelten Defizite aktuell in der SoC-Entwicklung bestehen und die Effizienz der Entwicklung beeinträchtigen. Da jedoch die Defizite *D3–D5* aus Sicht und Erfahrung der Befragten zu besonders umfangreichen Nacharbeiten führen können, ist deren Einfluss höher zu priorisieren.

Ergebnis und Diskussion: Ursachen

Im nächsten Schritt der Interviewevaluierung wurden die Teilnehmer/-innen zu den Ursachen für das jeweilige Defizit befragt. Dabei wurden ebenfalls Antwortmöglichkeiten als Teil des Leitfadens für die Priorisierung zur Verfügung gestellt, welche als mögliche Ursachen auf Basis der leitfadenfreien Interviews identifiziert wurden. Ziel dieser Befragung war es, festzustellen, ob der Ursprung der Defizite *D1–D5* auf eine defizitäre Entwicklungs- bzw. Spezifikationsmethodik zurückzuführen ist. Dabei soll der Einfluss der als Antwortmöglichkeiten bereitgestellten Ursachen auf die Entstehung des Defizits priorisiert werden. Die Priorisierung als „höchste Zustimmung“ (Priorisierungsstufe 1) entspricht somit der Einstufung als Hauptursache. Die Priorisierung als Priorisierungsstufe 5 („keine Zustimmung“) entspricht der Einschätzung, dass die Ursache nicht oder im Vergleich zu den übrigen Ursachen einen vernachlässigbaren Einfluss auf die Entstehung des Defizits hat.

Tabelle 6.3 zeigt die Fragestellung, die Antwortmöglichkeiten sowie die gZR als Ergebnis der Befragung bezüglich möglicher Ursachen für *D1*.

Tabelle 6.3 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 2.1

Fragestellung	Antwortmöglichkeiten	gZR	Kennung
Die hier gezeigten Ursachen *U1–U5* führen zum Defizit *D1* (Implementierungsfehler aufgrund unzureichender Spezifikation). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Unvollständige Spezifikation	*75 %*	*U1*
	Fehlende Eindeutigkeit der Spezifikation	*70 %*	*U2*
	Mangelnde Verständlichkeit/Lesbarkeit der Spezifikation	*15 %*	*U3*
	Fehlende Nachverfolgbarkeit der Anforderungen	*35 %*	*U4*
	Manueller Übertrag innerhalb der Spezifikationsdokumente sowie zwischen Spezifikation und Entwurf	*55 %*	*U5*

Die Interview-Teilnehmer/-innen wurden gebeten, die in der Tabelle 6.3 dargestellten Antwortmöglichkeiten zu nennen, welche aus ihrer Sicht als Ursache für das Defizit *D1* besteht. Da in der Regel mehrere Ursachen für die hier behandelten Defizite verantwortlich sind, wurden die Interview-Teilnehmer/-innen zudem gebeten, die Ursachen zu priorisieren. Nach der Befragung wurden die Antworten gemäß Abschnitt 6.1.2 gewichtet und die gZR berechnet.

Wie in der Tabelle 6.3 zu sehen, wurde die Unvollständigkeit der Spezifikation (*U1*) als Hauptursache für Implementierungsfehler mit einer gZR von 75 % priorisiert. So führt eine Unvollständigkeit der Spezifikation laut *SA2* und *SE1* dazu, dass die Modulentwickler/-innen die fehlenden Informationen durch Annahmen ergänzen und es in Folge dessen häufig zu Implementierungsfehlern kommen kann.

Die zweithöchste Priorisierung mit 70 % gZR bei der Interviewevaluierung erhielt die fehlende Eindeutigkeit der Spezifikation (*U2*) als Ursache für Implementierungsfehler. Ähnlich wie bei der Unvollständigkeit der Spezifikation sind auch hier die Entwickler/-innen gezwungen, die Informationen zu interpretieren und somit Annahmen zu treffen, welche zu Fehlern bei der Implementierung führen können.

Die *U3* (Mangelnde Verständlichkeit) mit 15 % gZR und *U4* (fehlende Nachverfolgbarkeit) mit 35 % gZR wurden von allen Interview-Teilnehmern/-innen gering priorisiert. Als Grund hierfür nannten die Interview-Teilnehmer/-innen *SE2* und *SE3*, dass sowohl die Verständlichkeit aus Sicht der Modulentwickler/-innen als auch die Nachverfolgbarkeit der Anforderungen bereits im aktuell angewendeten Entwicklungs-Flow gegeben sei.

Der manuelle Übertrag (*U5*) erhielt dabei von drei Interview-Teilnehmern/-innen die niedrigste und von vier Interview-Teilnehmern/-innen die höchste Priorisierung und kam damit auf ein gZR von 55 %. Hier lassen sich zwei Sichtweisen aus den Interviews extrahieren. Der manuelle Übertrag der Informationen zwischen Spezifikation und Entwurf führt häufig zu Fehlern bei der Implementierung und wurde daher vereinzelt hoch priorisiert. Dementgegen werden die Fehler bei der Verifikation entdeckt und verursachen somit meist nur geringe Auswirkungen auf die Effizienz der Entwicklung.

Das gleiche Vorgehen wurde für die Frage nach den Ursachen für *D2* gewählt. Den Interview-Teilnehmern/-innen wurden dieselben Antwortmöglichkeiten, wie in Tabelle 6.4 zu sehen, zur Verfügung gestellt. Erneut zeigt die Tabelle 6.4 die gZR als Ergebnis der Befragung.

Als Hauptursache mit 80 % gZR nannten die Interview-Teilnehmer/-innen eine unvollständige Spezifikation (*U1*). Eine unvollständige Spezifikation führt – wie bei der Implementierung – bei der Verifikation dazu, dass die Verifikationsingenieure/-innen zu Annahmen gezwungen sind, um die fehlenden Informationen zu ergänzen.

Tabelle 6.4 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 2.2

Fragestellung	Antwortmöglichkeiten	gZR	Kennung
Die hier gezeigten Ursachen *U1–U5* führen zum Defizit *D2* (Verifikationsfehler aufgrund unzureichender Spezifikation). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Unvollständige Spezifikation	*80 %*	*U1*
	Fehlende Eindeutigkeit der Spezifikation	*65 %*	*U2*
	Mangelnde Verständlichkeit/Lesbarkeit der Spezifikation	*38 %*	*U3*
	Fehlende Nachverfolgbarkeit der Anforderungen	*38 %*	*U4*
	Manueller Übertrag innerhalb der Spezifikationsdokumente sowie zwischen Spezifikation und Verifikation	*30 %*	*U5*

Dieser Sachverhalt birgt bei der Verifikation zudem die Gefahr, dass Fehler in der Implementierung durch eine defizitäre Verifikation nicht oder sehr spät erkannt und erhebliche Aufwände für die Fehlerbehebung verursacht werden. Ähnlich verhält es sich bei einer nicht eindeutigen Spezifikation (*U2*). Diese Ursache erhielt eine gZR von 65 % und birgt ebenfalls durch die Gefahr von Fehlinterpretationen das Risiko, dass die Implementierung nicht korrekt verifiziert wird und somit Fehler nicht erkannt werden können.

Die mangelnde Verständlichkeit (*U3*) und die fehlende Nachverfolgbarkeit der Anforderungen (*U4*) wurden mit einer gZR von 38 % priorisiert. Auch hier wurden die bestehende Verständlichkeit sowie die bestehende Nachverfolgbarkeit der Anforderungen als Begründung für die Priorisierung genannt. Darüber hinaus wurde die *U4* dreimal als höchst priorisierte und viermal als niedrig priorisierte Ursache genannt. In der weiteren Befragung zu den Gründen der Priorisierung wurden die unterschiedlichen Arbeitsweisen und Erfahrungen der Interview-Teilnehmer/-innen deutlich. Zum einen wurde die Ursache *U4* hoch priorisiert, da eine fehlende Nachverfolgbarkeit der Anforderungen aktuell zu erheblichen Problemen bei der Verifikation führt. So betonten alle Interview-Teilnehmer/-innen die Wichtigkeit der Nachverfolgbarkeit der Anforderungen. Darüber hinaus besteht laut *SE4* aktuell keine Zeit, um diese in den Projekten umzusetzen. Demgegenüber gaben die Befragten *SE2* und *SE3* an, dass die Nachverfolgbarkeit der Anforderungen in aktuellen Projekten bereits in hohem Maße gegeben sei und priorisierten daher *U4* niedrig ein.

Die geringste gZR mit 30 % erhielt der manuelle Übertrag als Ursache für Fehler bei der Verifikation. Zudem wurde hier seitens des *VI1* die aus seiner Sicht hohe Wichtigkeit einer manuellen Erstellung der Verifikation genannt, um diese erfolgreich durchführen zu können und die Übertragung von Fehlern aus der Spezifikation in die Verifikation zu verhindern.

Vergleichbar fiel die Priorisierung der Ursachen für nachträgliche Änderungen (*D3*) aus, wie Tabelle 6.5 zeigt.

Tabelle 6.5 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 2.3

Fragestellung	Antwortmöglichkeiten	gZR	Kennung
Die hier gezeigten Ursachen *U1–U5* führen zum Defizit *D3* (Nachträgliche Änderungen an Lasten- bzw. Pflichtenheft). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Unvollständige Spezifikation	*83 %*	*U1*
	Fehlende Eindeutigkeit der Spezifikation	*75 %*	*U2*
	Mangelnde Verständlichkeit/Lesbarkeit der Spezifikation	*53 %*	*U3*
	Fehlende Nachverfolgbarkeit der Anforderungen	*40 %*	*U4*
	Manueller Übertrag innerhalb der Spezifikationsdokumente sowie zwischen Spezifikation und Entwurf	*8 %*	*U5*

Auch hier wurde die Unvollständigkeit der Spezifikation (*U1*) mit 83 % gZR als Hauptursache für nachträgliche Änderungen genannt, da besonders eine fehlende Vollständigkeit in Pflichtenheft und Systemkonzept laut *SE1*, *SE3* und *SA1* zu einer fehlenden Realisierung von Funktionalitäten im SoC führen kann.

Vergleichbar verhält es sich bei der fehlenden Eindeutigkeit der Spezifikation (*U2*), da dies zu Fehlinterpretationen und damit gegebenenfalls sogar zu einer Realisierung falscher Funktionen führen könnte, welche so nicht vom Kunden gefordert wurden. *U2* wurde mit einer gZR von 75 % priorisiert.

Die mangelnde Verständlichkeit der Spezifikation (*U3*) wurde mit 53 % als Ursache für nachträgliche Änderungen priorisiert. Dabei birgt die mangelnde Verständlichkeit laut *SM1* besonders die Gefahr, dass der Kunde das Pflichtenheft bei der Planungsphase nicht korrekt versteht und somit ein fehlerbehaftetes Verständnis des Entwicklungsvorhabens erhält.

Die fehlende Nachverfolgbarkeit der Anforderungen (*U4*) wurde in diesem Zusammenhang oft als wichtig herausgestellt, da besonders die Nachverfolgbarkeit

zwischen Kundenanforderungen und Systemanforderungen helfen kann, nachträgliche Änderungen zu vermeiden. Die geringe Priorisierung von 40 % resultiert hierbei erneut aus der unterschiedlichen Einschätzung der Interview-Teilnehmer/-innen über die aktuell bestehende Nachverfolgbarkeit der Anforderungen in der SoC-Entwicklung. Eine sehr eindeutig niedrige Priorisierung mit 8 % gZR erhielt der manuelle Übertrag als Ursache für nachträgliche Änderungen.

Das Ergebnis der Befragung über die Ursachen für Inkonsistenzen Spezifikation/Entwurf sowie die Fragestellung und Antwortmöglichkeiten sind in Tabelle 6.6 dargestellt.

Tabelle 6.6 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 2.4

Fragestellung	Antwortmöglichkeiten	gZR	Kennung
Die hier gezeigten Ursachen *U1–U5* führen zum Defizit *D4* (Inkonsistenzen Spezifikation/Entwurf). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Unvollständige Spezifikation	*70 %*	*U1*
	Fehlende Eindeutigkeit der Spezifikation	*78 %*	*U2*
	Mangelnde Verständlichkeit/Lesbarkeit der Spezifikation	*18 %*	*U3*
	Fehlende Nachverfolgbarkeit der Anforderungen	*35 %*	*U4*
	Manueller Übertrag innerhalb der Spezifikationsdokumente sowie zwischen Spezifikation und Entwurf	*50 %*	*U5*

Bei der Frage nach den Ursachen für Inkonsistenzen zwischen den Spezifikationsdokumenten sowie zwischen Spezifikation und Entwurf (*D4*) wurde die fehlende Vollständigkeit (*U1*) als zweithöchste priorisierte Ursache mit 70 % gZR genannt. Die fehlende Eindeutigkeit der Spezifikation (*U2*) ist laut den Befragten mit 78 % gZR Hauptursache für Inkonsistenzen. Dabei wurde vermehrt von Situationen berichtet, bei denen in der Entwurfsphase der SoC auf Grundlage des Entwurfes weiterentwickelt wurde und nicht auf Grundlage der Spezifikation. Darüber hinaus wurden die so getroffenen Entwurfsentscheidungen nicht oder nur unzureichend in der Spezifikation nachgepflegt.

Die mangelnde Verständlichkeit der Spezifikation (*U3*) wurde mit einer gZR von 18 % niedrig priorisiert. Die mangelnde Verständlichkeit ist aus Sicht der Interview-Teilnehmer/-innen und damit aus Sicht der Entwickler/-innen eher selten der Fall im aktuell angewendeten Entwicklungs-Flow. Die fehlende Nachverfolgbarkeit der

Anforderungen (*U4*) wurde mit 35 % gZR priorisiert. Der manuelle Übertrag erhielt somit die dritthöchste Priorisierung mit 50 % gZR.

Bei der Befragung zu den möglichen Ursachen für die Fehlkommunikation zwischen Kunde und Entwickler wurden den Interview-Teilnehmern/-innen lediglich die Antwortmöglichkeiten Ursache *U1–U4* zur Verfügung gestellt, da die *U5* (manueller Übertrag) nicht als mögliche Ursache für die Fehlkommunikation mit dem Kunden infrage kommt (Tabelle 6.7).

Tabelle 6.7 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 2.5

Fragestellung	Antwortmöglichkeiten	gZR	Kennung
Die hier gezeigten Ursachen *U1–U5* führen zum Defizit *D5* (Fehlkommunikation Kunde-Entwickler). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Unvollständige Spezifikation	*65 %*	*U1*
	Fehlende Eindeutigkeit der Spezifikation	*80 %*	*U2*
	Mangelnde Verständlichkeit/Lesbarkeit der Spezifikation	*60 %*	*U3*
	Fehlende Nachverfolgbarkeit der Anforderungen	*35 %*	*U4*

Als Ursache für die Fehlkommunikation zwischen Kunde und Entwickler erhielt die unvollständige Spezifikation eine gZR von 65 % und ist damit die zweithöchst priorisierte Ursache für die Fehlkommunikation. Als Hauptursache mit 80 % gZR lässt sich die fehlende Eindeutigkeit der Spezifikation als Ergebnis der Befragung identifizieren. Auch die mangelnde Verständlichkeit erhielt in vielen Fällen eine hohe Priorisierung und kommt insgesamt auf 60 % gZR. Somit liegen die Ursachen *U1–U4* bei diesem Ergebnis besonders nah beieinander. Die fehlende Nachverfolgbarkeit erhielt als niedrig priorisierte Ursache 35 % gZR.

Zusammenfassend lassen sich als Hauptursachen für die hier betrachteten Defizite besonders die Unvollständigkeit und die fehlende Eindeutigkeit der Spezifikation herausstellen, da diese durchweg eine vergleichsweise hohe Priorisierung erhielten.

Ergebnis und Diskussion Lösungsansätze

Nach der Analyse zu möglichen Ursachen für die Defizite in der SoC-Entwicklung folgte die Befragung der Interview-Teilnehmer/-innen zu möglichen Lösungsansätzen, um die Defizite *D1* bis *D5* zu minimieren bzw. die Ursachen für die Defizite anzugehen. Wie bei den Fragen zuvor, standen auch hier den Interview-Teilnehmern/-innen unterschiedliche Antwortmöglichkeiten als Teil des Leitfadens

zur Verfügung. Dabei wurden die Interview-Teilnehmer/-innen erneut gebeten, die Lösungsansätze zu priorisieren. Hierbei bestanden einzelne der Antwortmöglichkeiten aus Teilaspekten der in dieser Arbeit entwickelten Gesamtmethode. So sind beispielsweise die Lösungsansätze „Modellbasierte Spezifikation“ und „Formalisierung Spezifikation“ Teil der hier vorliegenden Arbeit und wurden in Abschnitt 4.1 und Abschnitt 5.1 näher behandelt. Ebenso beinhaltet die Arbeit den „modellgetriebenen Entwurf“, eine „starke Nachverfolgbarkeit der Anforderungen“ und die „Arbeitsweise nach dem Top-down-Ansatz“. Die „Ausführbarkeit der Spezifikation“ ist jedoch in dieser Arbeit nur teilweise erfüllt und könnte daher in einer weiterführenden Arbeit Umsetzung finden.

Die Antworten wurden nach der Interviewevaluierung, wie zuvor in Abschnitt 6.1.2 beschrieben, quantifiziert und ein Mittelwert über alle Antworten der Interview-Teilnehmer/-innen gebildet. Tabelle 6.8 beinhaltet die Fragestellung sowie Antwortmöglichkeiten und das Ergebnis für die Frage 3.1. Die Fragestellung bleibt hierbei für alle Defizite gleich, die Antwortmöglichkeiten variieren jedoch.

Tabelle 6.8 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 3.1

Fragestellung	**Antwortmöglichkeiten**	**gZR**
Die hier angebotenen Lösungsansätze reduzieren bzw. beheben das Defizit D1 (Implementierungsfehler aufgrund unzureichender Spezifikation). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Modellbasierte Spezifikation	*78 %*
	Formalisierten Spezifikation	*75 %*
	Ausführbare Spezifikation	*28 %*
	Modellgetriebener Entwurf	*63 %*
	Stärkere Nachverfolgbarkeit der Anforderungen	*8 %*

Die höchste Priorisierung als möglichen Lösungsansatz für „Implementierungsfehler aufgrund einer unzureichenden Spezifikation“ wurde die modellbasierte Spezifikation mit 78 % gZR, dicht gefolgt von der formalisierten Spezifikation mit 75 % gZR genannt. Eine hohe gZR mit 63 % erhielt zudem der modellgetriebene Entwurf als Lösungsansatz. Die ausführbare Spezifikation mit 28 % und besonders die stärkere Nachverfolgbarkeit wurden bezogen auf das Vorbeugen von Implementierungsfehlern niedrig priorisiert.

Vergleicht man dieses Ergebnis mit den Ergebnissen aus der Befragung zu den Ursachen des Defizits *D1*, lässt sich eine Zuordnung ableiten. Als Hauptursachen wurden die Unvollständigkeit sowie die fehlende Eindeutigkeit der Spezifikation angegeben, als Lösungen für dasselbe Defizit die modellbasierte Spezifikation und

die formalisierte Spezifikation. Diese Lösungsansätze haben das Potenzial, die Vollständigkeit und Eindeutigkeit der Spezifikation signifikant zu erhöhen. Zu dieser Einschätzung kamen somit auch die Interview-Teilnehmer/-innen. Als dritte Ursache wurde der manuelle Übertrag priorisiert, und so liegt es nahe, dass auch die Automatisierung der Implementierung mittels modellgetriebenem Entwurf bei der Frage nach möglichen Lösungsansätzen hoch priorisiert wird.

Bei Frage 3.2 zu möglichen Lösungsansätzen und deren Wirkung für das Defizit *D2* (Verifikationsfehler aufgrund unzureichender Spezifikation) blieben die Antwortmöglichkeiten weitestgehend gleich. Lediglich der modellgetriebene Entwurf wurde durch die modellgetriebene Verifikation, welche ebenfalls Teil dieser Arbeit ist, ersetzt. Tabelle 6.9 zeigt die Fragestellung sowie die Antwortmöglichkeiten und das Ergebnis der Befragung zu Frage 3.2.

Tabelle 6.9 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 3.2

Fragestellung	Antwortmöglichkeiten	gZR
Die hier angebotenen Lösungsansätze reduzieren bzw. beheben das Defizit *D2* (Verifikationsfehler aufgrund unzureichender Spezifikation). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Modellbasierte Spezifikation	*53 %*
	Formalisierten Spezifikation	*75 %*
	Ausführbare Spezifikation	*39 %*
	Modellgetriebene Verifikation	*44 %*
	Stärkere Nachverfolgbarkeit der Anforderungen	*39 %*

Als möglicher Lösungsansatz für die Reduzierung von Verifikationsfehlern aufgrund einer unzureichenden Spezifikation wurde die Formalisierung der Spezifikation von den Interview-Teilnehmern/-innen mit 75 % gZR höchst priorisiert. Die Modellierung der Spezifikation mit 53 % gZR wurde als zweithöchste Priorisierung benannt. Die ausführbare Spezifikation (39 % gZR), die modellgetriebene Verifikation (44 % gZR) und die stärkere Nachverfolgbarkeit (39 % gZR) lagen nah beieinander und erhielten mit etwa 40 % ebenfalls eine verhältnismäßig hohe gZR.

Für die Befragung nach möglichen Lösungsansätzen, um nachträglichen Änderungen vorzubeugen, wurden die Antwortmöglichkeiten erneut entsprechend angepasst, wie in Tabelle 6.10 zu sehen. Die Tabelle 6.10 zeigt zudem die Fragestellung sowie das Ergebnis der Befragung in Form der gZR.

Bei der Frage, welche der hier gezeigten Lösungsansätze das Auftreten von nachträglichen Änderungen in der SoC-Entwicklung zu reduzieren vermag, wurde mit hoher Übereinstimmung der Top-down-Ansatz mit 85 % gZR höchst priorisiert. Durch die ausführliche Analyse der Kundenanforderungen und Anwendungsfälle

Tabelle 6.10 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 3.3

Fragestellung	Antwortmöglichkeiten	gZR
Die hier angebotenen Lösungsansätze reduzieren bzw. beheben das Defizit *D3* (Nachträgliche Änderungen an Lasten- bzw. Pflichtenheft). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Modellbasierte Spezifikation	*65 %*
	Formalisierte Spezifikation	*60 %*
	Ausführbare Spezifikation	*25 %*
	Top-down-Ansatz	*85 %*
	Stärkere Nachverfolgbarkeit der Anforderungen	*20 %*

sowie die lösungslosgelöste Dekomposition der benötigten Funktionen sinkt die Wahrscheinlichkeit, geforderte Funktionen zu vernachlässigen bzw. nicht geforderte Funktionen dennoch zu realisieren. Darüber hinaus steigt die Vollständigkeit der Spezifikation durch den Top-down-Ansatz in der Regel an.

Die modellbasierte Spezifikation mit 65 % gZR und die formalisierte Spezifikation mit 60 % gZR wurden ebenfalls als mögliche Lösungen hoch priorisiert. So sind sich die Interview-Teilnehmer/-innen einig, dass die Modellierung und Formalisierung der Spezifikation zu einer Steigerung der Spezifikationsqualität führt und somit nachträgliche Änderungen verhindert werden können. Die ausführbare Spezifikation (25 % gZR) und die stärkere Nachverfolgbarkeit der Anforderungen (20 % gZR) wurden verhältnismäßig niedrig priorisiert.

Das Ergebnis der Befragung zu möglichen Lösungsansätzen für die Reduzierung von Inkonsistenzen Spezifikation/Entwurf, die dabei verwendete Fragestellung sowie die Antwortmöglichkeiten sind nachfolgend in Tabelle 6.11 abgebildet.

Tabelle 6.11 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 3.4

Fragestellung	Antwortmöglichkeiten	gZR
Die hier angebotenen Lösungsansätze reduzieren bzw. beheben das Defizit *D4* (Inkonsistenzen Spezifikation/Entwurf). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Modellbasierte Spezifikation	*65 %*
	Formalisierte Spezifikation	*70 %*
	Ausführbare Spezifikation	*8 %*
	Modellgetriebener Entwurf	*55 %*
	Top-down-Ansatz	*53 %*

Um Inkonsistenzen zwischen Spezifikation und Entwurf vorzubeugen, wurde durch die Interview-Teilnehmer/-innen ebenfalls die Formalisierung der Spezifikation als höchst priorisierter Lösungsansatz mit 70 % gZR benannt. Dicht gefolgt

jedoch von der Modellierung der Spezifikation mit 65 % gZR, dem modellgetriebenen Entwurf mit 55 % gZR und dem Top-down-Ansatz mit 53 % gZR. Die gleichmäßige Verteilung könnte darauf hinweisen, dass ein kombinierter Ansatz, wie er in dieser Arbeit vorliegt, benötigt wird, um das Auftreten von Inkonsistenzen zu reduzieren. Als Ursache für das Defizit *D4* wurde die fehlende Eindeutigkeit, gefolgt von der Unvollständigkeit und dem manuellen Übertrag genannt. Diese Ursachen lassen sich am stärksten durch die Modellierung und Formalisierung der Spezifikation sowie durch die Automatisierung mittels modellgetriebenem Entwurf reduzieren. Die Ausführbarkeit der Spezifikation wurde in dieser Befragung sehr gering mit nur 8 % gZR priorisiert.

Tabelle 6.12 zeigt zum Schluss des Abschnittes die Fragestellung, mögliche Lösungsansätze sowie das Ergebnis der Befragung in Form der gZR. Befragt wurden die Interview-Teilnehmer/-innen zu möglichen Lösungsansätzen für die Reduzierung von Fehlkommunikationen zwischen Kunde und Entwickler sowie zwischen den Entwicklern/-innen.

Tabelle 6.12 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 3.5

Fragestellung	**Antwortmöglichkeiten**	**gZR**
Die hier angebotenen Lösungsansätze reduzieren bzw. beheben das Defizit *D5* (Fehlkommunikation Kunde-Entwickler). Inwieweit stimmen Sie den Antwortmöglichkeiten zu? Bitte priorisieren Sie.	Modellbasierte Spezifikation	*65 %*
	Formalisierte Spezifikation	*70 %*
	Ausführbare Spezifikation	*68 %*
	Stärkere Nachverfolgbarkeit Anforderungen	*48 %*

Wie zuvor kam auch bei dieser Frage eine nur knapp unterschiedliche Priorisierung zustande. Höchst priorisiert wurde die Formalisierung der Spezifikation mit 70 % gZR. Mit nur 2 % Unterschied folgte in diesem Fall die ausführbare Spezifikation mit 68 % gZR. Die modellbasierte Spezifikation erhielt eine gZR von 65 % und die stärkere Nachverfolgbarkeit der Anforderungen eine gZR von 48 %.

Dies ist die höchste Priorisierung der ausführbaren Spezifikation über das gesamte Evaluierungssinterview betrachtet. So sehen die Interview-Teilnehmer/-innen besonders bei der Kommunikation mit dem Kunden große Vorteile in der Ausführbarkeit der Spezifikation, um so die Verständlichkeit der Spezifikation neben der Formalisierung und Modellierung weiter zu erhöhen.

Bei der, im vorangegangenen Abschnitt beschriebenen Evaluierung möglicher Ursachen der nachfolgend näher betrachteten Defizite erhielten besonders die

Lösungsansätze Modellierung und Formalisierung der Spezifikation sowie Top-down-Ansatz eine durchweg hohe Priorisierung. Somit konnte der Nutzen der in dieser Arbeit vorgestellten modellbasierten Entwicklungsmethode auf die Reduzierung der Defizite und damit die Steigerung der Effizienz in der SoC-Entwicklung aus Sicht der Interview-Teilnehmer/-innen evaluiert und bestätigt werden.

Für die Kommunikation mit dem Kunden und um dabei Missverständnisse auszuschließen, sollte jedoch die Ausführbarkeit großer Teile der Spezifikation zusätzlich ermöglicht werden.

Bewertung der Aufwände

Zum Ende der Interviewevaluierung wurden die Teilnehmer/-innen gebeten, den Einfluss der neuen modellbasierten Methode auf die Aufwände in den nachfolgenden Bereichen im Vergleich zur aktuellen Arbeitsweise zu bewerten:

- Spezifikation
- Verifikation
- Entwurf
- SoC-Entwicklung gesamt

Zusätzlich wird die prozentuale Verringerung bzw. Erhöhung auf den durchschnittlichen Aufwand eines SoCs bezogen. Dazu wird ein gemittelter Wert von 20 Mio. Euro Entwicklungskosten für einen kundenspezifischen Automotive SoC als Grundlage genommen. Dieser Wert resultiert aus aktuellen Werten der SoC-Entwicklung der Firma *Robert Bosch GmbH*. Darüber hinaus teilen sich für die hier vorliegende Arbeit die Kosten wie folgt auf: Für die Spezifikation sind bei dem hier zugrunde liegenden Beispiel 3 Mio. Euro, für den Entwurf 5 Mio. Euro und für die Verifikation weitere 5 Mio. Euro vorgesehen. Die verbleibenden 7 Mio. Euro werden für Posten wie Projektplanung, Fehlerbehebung und Maskendruck benötigt. Aufgrund der großen Varianz bei der Komplexität und beim Umfang der SoCs handelt es sich hierbei lediglich um an die Realität angenäherte Werte. Neben den finanziellen Kosten wird eine durchschnittliche Entwicklungszeit von vier Jahren als Basis gewählt.

Tabelle 6.13 zeigt das Ergebnis der Befragung über den Einfluss der modellbasierten Entwicklung auf den Aufwand bei der Spezifikation. Dabei gaben alle Interview-Teilnehmer/-innen im ersten Schritt als Tendenz eine Erhöhung des Aufwandes an.

Die Tabelle 6.13 zeigt die Einstimmigkeit bezüglich der Tendenz aller Interview-Teilnehmer/-innen, dass sich der Aufwand im Bereich der Spezifikation durch Anwendung der neuen Methode auf die SoC-Entwicklung erhöhen wird. Dabei

Tabelle 6.13 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 4.1

Fragestellung	Antwortmöglichkeit	ZR
Wie wirkt sich die modellbasierte Entwicklungsmethode auf den Aufwand bei der Spezifikation, verglichen mit der aktuell angewendeten Arbeitsweise aus?	stark verringert (> 50 %)	*0 %*
	mittel bis stark verringert (25–50 %)	*0 %*
	leicht verringert (< 25 %)	*0 %*
	unverändert	*0 %*
	leicht erhöht (< 25 %)	*20 %*
	mittel bis stark erhöht (25–50 %)	*60 %*
	stark erhöht (> 50 %)	*20 %*

gaben die Interview-Teilnehmer/-innen *SA1*, *MA1*, *VI1* und *SE2* als Grund für diese Erhöhung eine vollständigere Spezifikation an. Das heißt, der Mehraufwand resultiert teilweise aus der Schließung der Lücken in der Spezifikation und ist daher für die Schaffung einer vollständigen Spezifikation in jedem Fall nötig.

60 % der Interview-Teilnehmer/-innen stimmten darüber hinaus einer mittleren Erhöhung des Aufwandes zu. Jeweils 20 % ZR erhielten eine leichte Erhöhung sowie eine starke Erhöhung des Aufwandes bei der Spezifikation eines SoCs. Erhöht sich der Aufwand bei der Spezifikation durch die Anwendung der Modellierungsmethode um 40 %, würde dies eine Steigerung der Kosten, bezogen auf das zu Beginn des Abschnittes genannte Beispiel für die SoC-Entwicklung, von etwa 3 Mio. auf etwa 4 Mio. Euro bedeuten. Somit konnte, aus Sicht der Befragten, durch die Anwendung der in dieser Arbeit entwickelten formalisierten Modellierungsmethode für die Spezifikation die gestellte Anforderung **A5** nicht erfüllt werden. Die Steigerung des Aufwandes bei der Spezifikation resultiert jedoch dabei zum Teil aus der Steigerung der Vollständigkeit bei der Spezifikation und lässt sich somit als nötiger Mehraufwand verstehen. Darüber hinaus ist eine Steigerung des Aufwandes bei der Spezifikation vertretbar, lässt sich doch als Resultat daraus der Aufwand in anderen Bereichen und besonders über die gesamte SoC-Entwicklung hinweg reduzieren.

Tabelle 6.14 zeigt das Ergebnis der Befragung über den Einfluss der modellbasierten Entwicklungsmethode auf den Aufwand beim Entwurf des SoCs verglichen mit der aktuell angewendeten Arbeitsweise. Hierbei gaben alle Interview-Teilnehmer/-innen als Tendenz eine Verringerung des Aufwandes an.

Einer leichten Verringerung des Aufwandes stimmten dabei 50 % der Interview-Teilnehmer/-innen zu. Weitere 30 % ZR erhielt die Verringerung im mittleren Bereich und 20 % ZR die Verringerung im starken Bereich. Bei einer Reduzierung des Aufwandes beim Systementwurf um 35 % würden die Kosten für den Entwurf bezogen auf den zu Beginn des Abschnittes gewählten Mittelwert von etwa 5 Mio.

Tabelle 6.14 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 4.2

Fragestellung	Antwortmöglichkeit	ZR
Wie wirkt sich die modellbasierte Entwicklungsmethode auf den Aufwand beim Entwurf verglichen mit der aktuell angewendeten Arbeitsweise aus?	stark verringert (> 50 %)	*20 %*
	mittel bis stark verringert (25–50 %)	*30 %*
	leicht verringert (< 25 %)	*50 %*
	unverändert	*0 %*
	leicht erhöht (< 25 %)	*0 %*
	mittel bis stark erhöht (25–50 %)	*0 %*
	stark erhöht (> 50 %)	*0 %*

auf etwa 3 Mio. Euro gesenkt werden. Als Gründe für die Verringerung wurden besonders die Reduzierung von Implementierungsfehlern aufgrund einer eindeutigeren und vollständigeren Spezifikation und die Reduzierung des Aufwandes durch den modellgetriebenen Entwurf genannt. Somit konnte, aus Sicht der Befragten, durch die hier vorgestellte modellbasierte Entwicklungsmethode der Aufwand beim Systementwurf reduziert und somit die Anforderung **A7** erfüllt werden.

Als Nächstes wurden die Interview-Teilnehmer/-innen zum Einfluss der Methode auf den Aufwand in der Verifikation befragt. Das Ergebnis der Befragung ist in Tabelle 6.15 als Diagramm dargestellt.

Tabelle 6.15 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 4.3

Fragestellung	Antwortmöglichkeiten	ZR
Wie wirkt sich die modellbasierte Entwicklungsmethode auf den Aufwand bei der Verifikation verglichen mit der aktuell angewendeten Arbeitsweise aus?	stark verringert (> 50 %)	*0 %*
	mittel bis stark verringert (25–50 %)	*30 %*
	leicht verringert (< 25 %)	*40 %*
	unverändert	*0 %*
	leicht erhöht (< 25 %)	*30 %*
	mittel bis stark erhöht (25–50 %)	*0 %*
	stark erhöht (> 50 %)	*0 %*

Bei dem Einfluss der modellbasierten Entwicklungsmethode dieser Arbeit auf den Aufwand bei der Verifikation gingen die Antworten bereits in Bezug auf die Tendenzen auseinander. So antworteten sieben Interview-Teilnehmer/-innen, dass aufgrund der erhöhten Eindeutigkeit, Vollständigkeit und Verständlichkeit der Spezifikation die Arbeit der Verifikationsingenieure/-innen erleichtert würde und somit

der Aufwand zwischen leicht (40 % ZR) und mittel (30 % ZR) sinke. Eine Reduzierung des Aufwandes bei der Verifikation von 30 % würde, bezogen auf das zu Beginn dieses Abschnittes genannten Mittelwerts eines SoCs, eine Reduktion von 5 Mio. auf circa 3,5 Mio. Euro bedeuten.

Drei der Interview-Teilnehmer/-innen, darunter der Interview-Teilnehmer VI1, gaben an, dass sich aufgrund der umfangreicheren und vollständigeren Spezifikation der Aufwand für die Verifikation erhöhe, gleichzeitig jedoch auch die Qualität zunehme. Somit resultiert die unterschiedliche Einschätzung des Einflusses auf den Aufwand in erster Linie aus der jeweiligen Sichtweise der Befragten. Die Reduzierung des Aufwandes bei der Verifikation wurde als Anforderung **A8** an die in dieser Arbeit entwickelte Entwicklungsmethode gestellt und kann laut 70 % der Interview-Teilnehmer/-innen erfüllt werden.

Tabelle 6.16 zeigt zum Ende des Abschnittes das Ergebnis der Interviewevaluierung – bezogen auf den Einfluss der modellbasierten Entwicklungsmethode – hinsichtlich des Aufwands über den gesamten SoC-Entwicklungs-Flow gesehen, da hierbei auch der Einfluss auf den Aufwand durch Nacharbeiten, Fehlkommunikation und Behebung von Fehlern betrachtet wird, welcher erst zu einem späteren Zeitpunkt der Entwicklung auftritt. Alle zehn Interview-Teilnehmer/-innen gaben als Tendenz eine Reduzierung des Aufwandes an.

Tabelle 6.16 Fragestellung, Antwortmöglichkeiten und Ergebnis Frage 4.4

Fragestellung	Antwortmöglichkeiten	ZR
Wie wirkt sich die modellbasierte Entwicklungsmethode auf den Aufwand bezogen auf die gesamte SoC-Entwicklung verglichen mit der aktuell angewendeten Arbeitsweise aus?	stark verringert (> 50 %)	*20 %*
	mittel bis stark verringert (25–50 %)	*80 %*
	leicht verringert (< 25 %)	*0 %*
	unverändert	*0 %*
	leicht erhöht (< 25 %)	*0 %*
	mittel bis stark erhöht (25–50 %)	*0 %*
	stark erhöht (> 50 %)	*0 %*

Darüber hinaus fiel das Ergebnis der Befragungen sehr eindeutig aus. So stimmten 80 % der Interview-Teilnehmer/-innen einer mittel bis starken Verringerung des Aufwandes über den gesamten SoC-Entwicklungs-Flow hinweg betrachtet zu. Weitere 20 % ZR erhielt der Skalenwert „stark verringert“. Sollte sich dieses Ergebnis auch auf Dauer in der SoC-Entwicklung erreichen lassen, könnte sich durch die in dieser Arbeit vorgestellte modellbasierte Entwicklungsmethode eine Reduzierung des Aufwandes von circa 40 % einstellen. Dies entspräche, bezogen auf die zu

Beginn des Abschnittes angegebenen Mittelwerte für die SoC-Entwicklung, einer Reduktion der Kosten von 20 Mio. Euro auf etwa 12 Mio. Euro sowie eine Reduktion der Entwicklungsdauer von vier auf zweieinhalb Jahre. Vergleicht man diesen reduzierten Wert der Kosten für die gesamte SoC-Entwicklung aus der Beispielrechnung mit der Summe der reduzierten Werte der Bereiche Spezifikation, Entwurf und Verifikation, fällt auf, dass die Summe der Einzelwerte mit 12,5 Mio. bereits den Wert der gesamten Kosten übersteigt. Darüber hinaus ist zu beachten, dass in den zu Beginn des Abschnitts angenommenen Kosten für die gesamte SoC-Entwicklung (20 Mio. Euro) bereits die „Sonstigen Kosten“ (7 Mio. Euro) stellvertretend für die Bereiche Projektplanung, Fehlerhebung und Maskendruck enthalten sind.

Wird der Einfluss der modellbasierten Entwicklungsmethode auf die gesamten Kosten der SoC-Entwicklung betrachtet, resultiert die starke Senkung der Kosten, so *MA1,* neben der Reduktion der Kosten bei Entwurf und Verifikation besonders aus der Reduzierung der Kosten bei der Fehlerbehebung und aus der Vermeidung von Rekursionen bei der Maskenherstellung des Wafers. Muss diese Maskenerstellung aufgrund von Fehlern oder nachträglichen Anpassungen wiederholt werden, entsteht laut *MA1* eine Verzögerung von etwa einem Jahr und etwa 1 Mio. Euro an Mehrkosten pro Rekursion.

Dabei bleibt jedoch in diesem Abschnitt die Zunahme der Komplexität der zukünftig zu entwickelnden SoCs, welche der Reduktion des Aufwandes entgegenwirken würde, unbeachtet. Dennoch sehen, so das Ergebnis der Interviewevaluierung, alle Interview-Teilnehmer/-innen die Reduzierung des Aufwandes, bezogen auf die gesamte SoC-Entwicklung, durch Anwendung der modellbasierten Entwicklungsmethode als erfüllt an. Somit kann das in Kapitel 1 beschriebene übergeordnete Ziel der Effizienzsteigerung als erfüllt angesehen werden.

6.2 Evaluierung modellgetriebener Systementwurf

Wie bereits in der vorangegangen Arbeit [76] gezeigt wurde, kann bereits durch die teilweise Automatisierung des Entwurfs eine erhebliche Reduktion des Aufwandes erreicht werden. Dafür sind jedoch die Qualität und die Korrektheit des Generierungsergebnisses entscheidend. Aus diesem Grund wurde als Teil der hier vorliegenden Arbeit zusätzlich zu einer Evaluierung der Methode durch Interviews eine Verifikation des generierten Codes durchgeführt.

Co-Verifikation sowie Co-Simulation beschreiben die kombinierte Simulation von Hardware und Software. In der hier vorliegenden Arbeit wurde somit die

generierte Software nebenläufig zur generierten Hardware in Form des Virtuellen Prototyps simuliert und verifiziert. Der Virtuelle Prototyp dient somit bei der Simulation als Hardware-Emulator, welcher noch nicht der finalen Hardware des zu entwickelnden Systems entspricht, dabei jedoch das Verhalten der Hardware nachbildet. Für die Co-Simulation wurde dabei nachfolgend beschriebene Simulationsanordnung verwendet.

6.2.1 Simulationsanordnung

Für die Simulation wurde die in Abbildung 6.3 gezeigte Simulationsanordnung verwendet. Dabei zeigt die Abbildung 6.3, welche Tools für die Simulation des Virtuellen Prototyps in Verbindung mit der Software benötigt werden.

- Die zu simulierende Software wird mit **µVision Keil** kompiliert.
- Die kompilierte Software wird von dem in Abschnitt 5.3 beschriebenen Tool **Virtualizer Studio** geladen und ausgeführt.
- Das Debuggen der Software und die damit verbundene Steuerung der Simulation erfolgen mithilfe einer **Eclipse-„Integrated Development Environment**“ (IDE, Integrierte Entwicklungsumgebung).

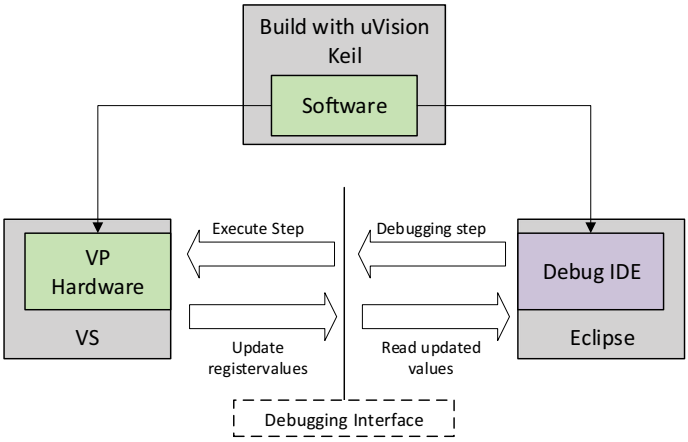

Abbildung 6.3 Simulationsanordnung Virtueller Prototyp

Dieser Ablauf gewährleistet, dass die Debugging Session und die SystemC-Simulation synchronisiert sind und das Debugging die Steuerung der Gesamtsimulation übernimmt. Während der laufenden Simulation können im Virtualizer Studio alle Registerwerte der Systemkomponenten und die Speicherinhalte ausgelesen und manipuliert werden.

6.2.2 Co-Verifikation

Die Co-Verifikation des generierten Codes erfolgt mittels dynamischen Testens und umfasst acht Testszenarien, welche bei den nachfolgenden Kombinationen zwischen Virtuellem Prototyp und Software angewandt wurden:

- Verifikation der generierten Software in Kombination mit manuell implementiertem und bereits verifiziertem Code des Virtuellen Prototyps
- Verifikation des generierten Codes des Virtuellen Prototyps in Kombination mit manuell implementiertem und verifiziertem Softwarecode
- Verifikation der Kombination aus generiertem Softwarecode und generiertem Code des Virtuellen Prototyps

Nachfolgend wird die Simulation der Testfälle anhand der Bootloading-Funktionalität beschrieben, welche Teil des in Abschnitt 5.2 vorgestellten *Processor-Subsystem* ist. Als Teil dieser Arbeit wurde die Bootloading-Funktionalität, wie in Abschnitt 5.2 erläutert, in SysML modelliert und der entsprechende C-Code mittels des in Abschnitt 5.3.3 beschriebenen Vorgehens generiert. Zusätzlich wurde die Architektur- wie auch die SystemC- Verhaltensbeschreibung der Hardware des Virtuellen Prototyps aus dem SysML-Modell generiert.

Der Bootloader ist das erste Programm, welches bei der Initialisierung eines SoCs ausgeführt wird. Eine der Hauptfunktionen des Bootloadings ist das Schreiben von Initialwerten aus dem *OTP* in die verschiedenen Register der Module durch das *CRC_module* beim Start des SoCs.

Bei der Durchführung der Co-Simulation wurden acht verschiedene Testszenarien ausgeführt. Jedes der Testszenarien **T1** bis **T8** beinhaltet einen definierten Initialwert im OTP (One-Time-Programmable), welcher zu Beginn der Simulation geladen werden muss. Diese Initialwerte werden bei der Ausführung des Bootloadings ausgelesen und an eine vordefinierte Adresse des Systems geschrieben. Die Adressen der unterschiedlichen Testszenarien sind sowohl RAM-Adressen als auch Konfigurationsregister einzelner Hardware-Komponenten. Das Bootloading

wird bei der Simulation jedes einzelnen Testszenarios vollständig ausgeführt, und anschließend werden die Registerwerte und der RAM ausgelesen. Die so ausgelesenen Werte und zugehörigen Adressen werden mit den Erwartungswerten aus der Testspezifikation der jeweiligen Testszenarien verglichen und verifiziert. Dieses Vorgehen entspricht dem White-Box-Prinzip [17].

Das Beispiel des Bootloadings ist in diesem Zusammenhang besonders aussagekräftig, da bei dessen Ausführung die kontrollflussorientierten Module des Virtuellen Prototyps von der Software angesteuert werden. Dabei wird somit die gesamte in Abbildung 5.11 gezeigte Architektur des Beispiel-SoCs wie auch die Software bei der Co-Simulation genutzt und getestet.

Nach der Verifikation des korrekten Verhaltens sowohl der generierten Software als auch des generierten Virtuellen Prototyps für das Bootloading wurde die Co-Simulation mit den Testszenarien **T1** bis **T8** ein weiteres Mal wiederholt, um das Verhalten der Software im Fehlerfall zu verifizieren. So müssen beispielsweise Fehler bei der Übertragung der Initialwerte durch das *CRC Module* mittels Redundanzcheck erkannt und korrigiert werden. Daher wurden zusätzliche Testfälle entwickelt, welche Fehler bei der Durchführung des Bootloadings provozieren und die Korrektheit des Verhaltens im Fehlerfall verifizieren. Dabei ist entscheidend, dass der fehlerhaft übertragene und anschließend geschriebene Initialwert im RAM beziehungsweise im Konfigurationsregister der Hardware-Komponenten dem zu erwartenden Wert der Testspezifikation entspricht.

Durch das hier beschriebene Vorgehen für die Co-Simulation mittels dynamischen Testens wurde versucht, durch die Testszenarien **T1** bis **T8** alle möglichen Verzweigungen der Software zu simulieren und zu prüfen, um so eine möglichst hohe Testabdeckung (Code Coverage) zu erreichen.

Testergebnisse

Als Ergebnis der in dieser Arbeit durchgeführten und in den vorangegangenen Abschnitten beschriebenen Tests lässt sich die Abwesenheit von Fehlern sowie die Abwesenheit von funktionellen Unterschieden zwischen generiertem und manuell implementiertem Code belegen. Dies gilt sowohl für die generierten Anteile der Software als auch für die in SystemC generierten kontrollflussorientierten Module des Virtuellen Prototyps. Zudem wurde ein korrektes Verhalten im Fehlerfall nachgewiesen. Jedoch geben die Tests keine Informationen über die Unterschiede in der Performance zwischen generiertem und manuell implementiertem Code.

6.2.3 Performance-Tests

Neben der Verifikation der korrekten Funktionalität der generierten Software und des generierten Virtuellen Prototyps im Vergleich zum manuell implementierten Code sollte die Performance des jeweiligen Codes verglichen werden. Dazu wurde im ersten Schritt die Laufzeit des manuell implementierten Codes und des generierten Codes bei der Ausführung der im vorangegangenen Abschnitt beschriebenen Testszenarien **T1** bis **T8** gemessen und verglichen. Tabelle 6.17 zeigt das Ergebnis der Messung.

Tabelle 6.17 Performance-Test: Laufzeit

Testszenarien	**Laufzeit**		
	Manuell Implementierter Code [µs]	***Generierter Code [µs]***	***Abweichung in %***
Testszenario T1	37,262	37,223	0,001
Testszenario T2	37,457	37,340	0,003
Testszenario T3	24,410	24,293	0,005
Testszenario T4	19,122	19,396	−0,014
Testszenario T5	19,982	20,060	−0,004
Testszenario T6	21,085	21,202	−0,006
Testszenario T7	21,124	21,202	−0,004
Testszenario T8	30,480	30,519	−0,001

Die Tabelle 6.17 zeigt die ermittelte Laufzeit des manuell implementierten Codes im Vergleich zum generierten Code, gemessen in Mikrosekunden. In der vierten Spalte wurde die Differenz zwischen den beiden Werten berechnet und prozentual dargestellt. Die gemessenen Laufzeiten weisen dabei nur minimale Unterschiede auf. Diese lassen sich mit hoher Wahrscheinlichkeit auf andere Parameter zurückführen, welche zu üblichen Schwankungen bei der Simulationsgeschwindigkeit führen können. Somit lässt sich für die generierten Funktionen, wie zu erwarten, eine vergleichbare Performance bezogen auf die Laufzeit nachweisen.

Zusätzlich zur Laufzeit wurde im nächsten Schritt der Speicherbedarf der Bootloading-Teilfunktionen im ROM gemessen und verglichen. Die jeweilige Software-Funktion wurde dazu kompiliert und der ROM-Verbrauch der drei

internen Bootloading-Funktionen ausgelesen und verglichen. Das Ergebnis des Vergleichs wird in Tabelle 6.18 veranschaulicht.

Tabelle 6.18 Performance-Test: Speicherbedarf

Bootloading Interne Funktion	**ROM-Speicherbedarf**		
	Manuell Implementierter Code [Byte]	***Generierter Code [Byte]***	***Abweichung in %***
Teilfunktion TF1	107,42	107,42	0,00 %
Teilfunktion TF2	642,58	636,72	0,91 %
Teilfunktion TF3	332,03	330,07	0,59 %

Die Tabelle 6.18 zeigt den ermittelten Speicherverbrauch des manuell implementierten Codes der drei Bootloading-Teilfunktionen verglichen mit dem jeweiligen generierten Code in Byte. In der vierten Spalte ist die berechnete prozentuale Abweichung dargestellt. Bei der Funktion 1 wurde, wie in der Tabelle 6.18 zu sehen ist, kein Unterschied beim Speicherverbrauch gemessen. Im Gegensatz dazu zeigen die Funktionen 2 und 3 leichte Unterschiede. Dies lässt sich auf kleinere Unterschiede im generierten Code verglichen mit dem implementierten Code zurückführen, welche die Funktionalität des Codes jedoch nicht beeinflussen. So wurde beispielsweise bei der manuellen Implementierung eine Switch-Case-Anweisung implementiert, während der Generator dasselbe Verhalten als If-Else-If-Verkettung generiert. Interessant ist jedoch, dass die generierten Teilfunktionen 2 und Teilfunktion 3 einen geringeren Wert für den Speicherverbrauch als der manuell implementierte Code aufweisen. Da die Unterschiede jedoch lediglich gering sind, lässt sich auch hier – verglichen mit dem manuell implementierten Code – eine vergleichbare Performance bezogen auf den Speicherverbrauch des generierten Codes nachweisen.

6.3 Zwischenergebnis

In dem vorangegangenen Kapitel konnte, mittels Interviewevaluierung, die Reduzierung der diskutierten Defizite durch die Anwendung der in dieser Arbeit entwickelten modellbasierten Entwicklungsmethode evaluiert werden. Dabei wurden neben der Analyse der Ursachen für das jeweilige Defizit mögliche Lösungsansätze für die Reduzierung des Defizites analysiert. Es konnte somit nachgewiesen

werden, dass auf Basis der Erfahrungen der Befragten die in dieser Arbeit vorgestellte modellbasierte Entwicklungsmethode die priorisierten Lösungsansätze für die Reduktion des Defizites enthält und somit in der Lage ist, die Effizienz der Entwicklung zu erhöhen. Darüber hinaus wurden die Interview-Teilnehmer/-innen zu dem Einfluss der modellbasierten Entwicklungsmethode auf den Aufwand in den Bereichen Spezifikation, Entwurf, Verifikation und in dem gesamten SoC-Entwicklungs-Flow befragt. Hierbei konnte festgestellt werden, dass zwar der Aufwand bei der Spezifikation steigt, der Aufwand der anderen Bereiche und der gesamten SoC-Entwicklung durch die Reduzierung von auftretenden Defiziten und die Verbesserung der Methodik jedoch erheblichen verringert werden kann.

Neben der Evaluierung der Methode wurde durch die Co-Verifikation und die Durchführung verschiedener Performance-Tests die Qualität und Korrektheit des generierten Codes sowohl im Bereich der Software als auch im Bereich des Virtuellen Prototyps nachgewiesen.

In Kapitel 7 folgt die Zusammenfassung der Arbeit sowie ein Ausblick möglicher zukünftiger Themen für weiterführende Arbeiten auf dem Gebiet der modellbasierten Entwicklungsmethoden.