Evaluation der domänenspezifischen Sprache HMISL zur modellgetriebenen Entwicklung von Automotive HMIs

Simon Gerlach

HMI-Systemtechnik
Volkswagen AG
Brieffach 1588
D-38436 Wolfsburg
simon.gerlach@volkswagen.de

Abstract: HMISL ist eine domänenspezifische Sprache (DSL) zur Entwicklung der Verhaltenslogik von Benutzerschnittstellen von Infotainmentsystemen, die als Ergänzung marktverfügbarer GUI-Entwicklungswerkzeuge eingesetzt werden soll. Die HMISL wurde hinsichtlich Tauglichkeit, Erlernbarkeit und Verständlichkeit evaluiert und ermittelt, ob sich durch ihren Einsatz Effizienzgewinne gegenüber bisherigen Vorgehensweisen erzielen lassen. Dazu wurde die Sprache eingesetzt, um einen Teil der Verhaltenslogik eines aktuellen Infotainment-HMIs zu implementieren. Außerdem wurde eine Nutzerstudie mit erfahrenen Entwicklern aus der Zielgruppe durchgeführt. Darin wurde gezeigt, dass die grundsätzlichen Programmierparadigmen der HMISL für ihre Zielgruppe leicht verständlich sind. Analytische Aufgabenstellungen konnten von den Probanden ohne Probleme gelöst werden, bei deren praktischen Durchführung wurde jedoch deutlich, dass bestimmte Verbesserungen an der Syntax der HMISL sinnvoll sind.

1 Hintergrund

Ziel der modellgetriebenen Softwareentwicklung (MDSD) ist es, die finale Software automatisch aus formalen Modellen zu erzeugen. Dies erfordert das Vorhandensein entsprechender Modelle, in denen alle dafür erforderlichen Informationen ausgedrückt werden können. Für die modellgetriebene Entwicklung der Benutzerschnittstellen (HMI) von Fahrzeug-Infotainmentsystemen existiert eine Vielzahl unterschiedlicher Modellierungswerkzeuge am Markt. Beispiele dafür sind Elektrobit GUIDE, Mecel Populus und Fujitsu CGI Studio. Diese Werkzeuge unterscheiden sich stark hinsichtlich der damit realisierbaren HMIs. So kann es zur Entwicklung neuartigen Bedienkonzepte erforderlich sein, dreidimensionale Szenen zu rendern und die Form oder Oberflächenbeschaffenheit der darin enthaltenen Körper zur Laufzeit zu manipulieren. Dies erfordert den Einsatz anderer Werkzeuge als bei konventionellen 2D-HMIs, in denen die grafische Darstellung zumeist aus Kompositionen vorgerenderter Bilder zusammengesetzt werden kann. Es ist daher erforderlich, anhand der projektspezifischen Anforderungen das jeweils am besten geeignete Werkzeug zur Entwicklung des HMIs auswählen zu können.

Die verfügbaren Werkzeuge unterscheiden sich im Funktionsumfang und setzen unterschiedliche Modellierungsmittel ein. Zumeist fokussieren sie damit jedoch die strukturellen Teile des HMIs und insbesondere das GUI-Layout. Ein vollständiges HMI benötigt jedoch zudem die Realisierung des Dialogflusses, der Steuerung von Animationen, Verarbeitung von Benutzereingaben und daraufhin den Aufruf von Gerätefunktionen sowie die Aufbereitung von Daten zur Anzeige. Dieses dynamische Verhalten des HMIs kann dagegen mit den Werkzeugen nicht vollständig modelliert werden. Teilweise stehen dafür Statecharts zur Verfügung, komplexe algorithmische Berechnungen oder String-Formatierungen sind damit jedoch kaum auszudrücken.

Für diesen Teil der Software ist somit mit den vorhandenen Werkzeugen eine modellgetriebene Entwicklung nicht möglich, stattdessen muss er manuell implementiert werden. Bei der Arbeitsteilung zwischen OEM und Zulieferer [Ger11a] kann dies dazu führen, dass auf Seiten des OEMs beide Entwicklungsmethoden gleichzeitig zum Einsatz kommen und miteinander synchronisiert werden müssen, was zu zahlreichen Problemen führt [Ger11b]. Ein wichtiger Grund dafür ist, dass die komplexen Modellierungswerkzeuge nur von speziell geschulten Experten genutzt werden können, während die Entwicklung der Verhaltenslogik durch Softwareentwickler vorgenommen wird. Diese Aufgabenteilung verhindert eine vertikale, anforderungsorientierte Entwicklung und führt zu einem hohen Kommunikationsaufwand. Außerdem müssen Kompromisse in der Architektur der Software eingegangen werden, um den generierten und den manuell implementierten Code zu integrieren und die referentielle Integrität überprüfen zu können. Infolgedessen kann die Codemenge wachsen oder die Performance nicht optimal ausgeschöpft werden. Ein Beispiel dafür ist die Erzeugung von Proxy-Komponenten, um generierte Elemente unter verständlichen Bezeichnern im manuell implementierten Code anzusprechen. Letztlich ist auch die Fehlerbehebung erschwert, weil dabei sowohl der generierte wie auch der handgeschriebene Code gemeinsam betrachtet und ihr Zusammenspiel verstanden werden muss.

Aus diesem Grund wurde die HMISL (HMI Specification Language) entwickelt, eine domänenspezifische Sprache (DSL) zur Ergänzung der marktverfügbaren HMI-Modellierungswerkzeuge. Die HMISL ist mit unterschiedlichen Modellierungswerkzeugen einsetzbar, damit diese anhand der projektspezifischen Anforderungen ausgewählt werden können. Primäre Aufgabe der HMISL ist die plattformunabhängige Modellierung der Verhaltenslogik des HMIs, um so eine vollständige modellgetriebene Entwicklung der HMI-Software zu ermöglichen. Darüber hinaus beinhaltet sie Konzepte zur Bildung verschiedener HMI-Varianten für unterschiedliche Funktionsumfänge, Marken, Fahrzeugmodelle und Landesprachen. Um eine einfache Erlernbarkeit zu erzielen, verwendet die HMISL das bekannte Modellierungskonstrukt der Statecharts, setzt dafür jedoch eine textuelle Notation ein. Sie ergänzt dies mit einer einfachen imperativen und statisch-typisierten Programmiersprache, deren Syntax an Java angelehnt ist. Dadurch sollen auch diejenigen Teile der Verhaltenslogik formuliert werden können, für die bisher konventionelle Programmiersprachen eingesetzt wurden. Ausführlichere Informationen zu den Konzepten der HMISL finden sich in [Ger11b] und [Ger11c].

2 Ziele der Evaluation

Um die Eignung der zuvor beschriebenen Konzepte zu evaluieren muss nachgewiesen werden, dass die HMISL die erforderlichen Mittel zur Implementierung der Verhaltenslogik aufweist und ihr Einsatz Effizienzgewinne gegenüber den bisherigen Entwicklungsprozessen bietet. Zudem soll geprüft werden, ob die HMISL für ihre potentiellen Nutzer einfach verständlich und schnell erlernbar ist.

3 Prototyping

DSLs wurden bereits in der Praxis für verschiedene Zwecke eingesetzt und dabei ihr Nutzen nachgewiesen [HB88] [KKB96]. Tolvanen berichtet beispielsweise von industriellen Case-Studies, in denen Produktivitätssteigerungen um den Faktor 5-10 erzielt wurden [Tol05]. Da DSLs naturgemäß domänenspezifisch sind, können diese Ergebnisse nicht auf andere Einsatzbereiche übertragen werden. Völter empfiehlt stattdessen die Tauglichkeit eines solchen Ansatzes durch die Entwicklung eines Prototyps nachzuweisen [Voe11]. Im Falle der HMISL ist Tauglichkeit dann gegeben, wenn sie die erforderlichen Sprachmittel bietet, um das Verhalten eines Infotainmentsystems anforderungsgemäß zu implementieren. Um dies nachzuweisen, soll im Prototyp eine Teilmenge davon umgesetzt werden, die so gewählt ist, dass darin alle relevanten Implementierungskonzepte enthalten sind. Zudem muss ihr Erstellungsaufwand beschränkt sein, gleichzeitig aber eine realistische Komplexität der Modelle erreicht werden und alle Bestandteile der HMISL zum Einsatz kommen.

Als Vorbereitung wurde dafür aus der Software eines aktuellen Infotainmentsystems der gewählte Teil der Verhaltenslogik entfernt und anschließend mit Hilfe der HMISL erneut implementiert, ausgehend von den ursprünglichen Anforderungsdokumenten. Damit sich ein lauffähiges Gesamtsystem ergibt, wurden die Codegeneratoren der HMISL so angepasst, dass sie Code für die Architektur im Zielsystem erzeugen.

Zur Entwicklung des Prototyps wurden weitere Sprachmittel erforderlich, die unmittelbar in der HMISL ergänzt werden konnten. Dadurch wurde am Ende eine Ausdrucksmächtigkeit erreicht, mit der eine vollständige Umsetzung aller gewählten Anforderungen möglich war. Ein zu ergänzendes Sprachmittel waren Annotationen, um dem Codegenerator Informationen über Sonderbehandlungen mitzuteilen. Dies wurde benötigt, um auch dann kompilierfähigen und ausführbaren Code zu erzeugen, wenn dieser handgeschriebenen Code anspricht, in dem unterschiedliche Implementierungen für gleiche Sachverhalte verwendet werden. Außerdem war es erforderlich, Bezeichner verwenden zu können, die bereits als reserviertes Wort verwendet wurden. Dafür wurde ein Escape-Symbol eingeführt, das in einem solchen Fall vorangestellt werden kann.

Es ist offensichtlich, dass man bei einem Praxiseinsatz weitere Sprachmittel ergänzen muss, beispielsweise für Sonderaufgaben oder für Spezifika bestimmter Zielframeworks. Um die HMISL auch zur Weiterentwicklung von Systemen einzusetzen, die bisher konventionell implementiert worden sind, müssen zudem Strategien zur Einbindung vorhandener Codeteile erstellt werden.

Während der Erstellung des Prototyps wurde auch eine Zeitmessung durchgeführt. Zur Ermittlung der Effizienz ist es jedoch nicht zielführend, die erforderlichen Zeiten zur Implementierung vorgegebener Anforderungen mit der HMISL und der bisher praktizierten Vorgehensweise zu vergleichen. Der wichtigste Grund dafür ist, dass während der ursprünglichen Implementierung die Anforderungen noch Änderungen unterworfen waren, diese jedoch bei der Reimplementierung endgültig und vollständig vorlagen. Problematisch ist auch, dass es wegen der unterschiedlichen Komplexität des Prototyps und des realen Systems zu Skalen- und Lerneffekten kommen kann. Obgleich Bettin [Bet04] glaubt, den Erfolg von MDSD in der Kompaktheit der Modelle messen zu können, ist zur Effizienzbewertung ein Vergleich zwischen LOC der HMISL-Modelle und des dadurch ersetzten Code ungeeignet. Hierbei würde nicht berücksichtigt, dass die Komplexität der Modelleerstellung höher sein kann als die der gleichen Menge Code. Wegen unterschiedlicher Programmierparadigma ist ein Vergleich anderer Codemetriken[1] ebenso nicht sinnvoll.

4 Nutzerstudie

Mit Hilfe des Prototyps konnte zuvor die Eignung der HMISL für den Einsatzzweck nachgewiesen werden. Durch Ergänzen weiterer Sprachmittel wurde eine Ausdrucksmächtigkeit erreicht, um die vorgegebenen Anforderungen umsetzen zu können. Zur Beantwortung der übrigen Forschungsfragen wurde anschließend eine Nutzerstudie durchgeführt. Der bestehende Entwicklungsprozess für die HMI-Software des Infotainmentsystems ist jedoch zu komplex, um Vergleichswerte als Referenzsystem bestimmen zu können. Somit können zwar Zusammenhänge innerhalb der quantifizierbar erfassbaren Daten ermittelt und Verbesserungspotentiale aufgedeckt werden, die Forschungsfragen können jedoch nur qualitativ beantwortet werden.

4.1 Bewertungskriterien

Um messbare Akzeptanzkriterien zu erhalten, wurden die Forschungsfragen präzisiert:

- *Tauglichkeit:* Nachdem Nutzer die Sprachkonzepte verwendet haben, sollen sie deren Eignung für reale Serienprojekte einschätzen. Dabei sollen mindestens 80% angeben, dass sich diese „gut“ oder „sehr gut“ dafür einsetzen lassen.
- *Erlernbarkeit:* Nach 30 minütiger Schulung sollen mindestens 80% der Nutzer eine vorgegebene Aufgabenstellung mit Hilfe der HMISL umsetzen können.
- *Verständlichkeit:* Nach 30 minutiger Schulung sollen mindestens 80% der Nutzer an einem vorhandenen Modell Erweiterungen und Fehlerbehebungen vornehmen können.

[1] Beispielsweise wird von da Silva die McCabe-Komplexität eines Modells in UMLi mit der eines entsprechenden Modells in UML verglichen, um so die größere Einfachheit und Kompaktheit nachzuweisen [SP03]. Dabei wird jedoch nicht berücksichtigt, dass die Komplexität der Modelle auch wesentlich von den Fähigkeiten der Modellierer in der jeweiligen Sprache abhängt.

- *Effizienzgewinne:* Nachdem Nutzer die HMISL praktisch eingesetzt haben, soll die Mehrzahl von ihnen überzeugt sein, dass ihr Einsatz effizienter ist als die bisher praktizierte Vorgehensweise.

Nutzer sind hierbei stets Software-Entwickler für Automotive-HMI-Systeme mit mindestens 18 Monaten Berufserfahrung in Vollzeit, denen die HMISL bisher unbekannt ist. Sie müssen davon mindestens 1 Jahr in einem Tätigkeitsschwerpunkt arbeiten, der im potentiellen Einsatzbereich der HMISL liegt. Dadurch kann ein fundiertes Wissen über die Einsatzdomäne vorausgesetzt werden und ihnen ist das Referenzsystem ausreichend bekannt. Die Probanden sollen zufällig ausgewählt werden, wobei jedoch sichergestellt werden muss, dass alle Aufgabengebiete in repräsentativen Anteilen vertreten sind.

4.2 Methodik

Aus der Gruppe der Serienentwickler eines Infotainmentsystems wurden Probanden anhand der zuvor genannten Kriterien ausgewählt und zu der Studie eingeladen. Die jeweils 90-minütigen Untersuchungen fanden einzeln und in einem separaten Untersuchungsraum statt. Dabei waren der Versuchsleiter und ein Protokollführer anwesend, letzterer war an der Entwicklung der HMISL nicht beteiligt. Die Beteiligung an der Studie fand während der Arbeitszeit statt und wurde nicht zusätzlich vergütet.

Jede Untersuchung hatte den gleichen Ablauf. Zunächst wurde den Probanden der Versuchsablauf erläutert. Dann wurden mittels eines sozio-demografischen Fragebogens grundlegende Daten des Probanden erhoben und er wurde gebeten, eine Selbsteinschätzung seiner Expertise vorzunehmen. Außerdem sollte er mehrere Aussagen zu MDSD im Allgemeinen und zu dessen Einsatz für Automotive-HMIs bewerten. Danach wurden dem Probanden die Konzepte der HMISL über eine standardisierte Präsentation erläutert. Dabei waren Rückfragen erlaubt, die Gesamtdauer war jedoch auf 30 Minuten beschränkt. Den Probanden standen die Präsentationsfolien sowie Papier und Stift für Notizen zur Verfügung. Zudem wurde ihnen eine Übersichtskarte mit den wichtigsten Sprachelementen der HMISL ausgeteilt.

Anschließend wurden die Entwickler mit mehreren Aufgabenstellungen konfrontiert, die mit Hilfe der HMISL gelöst werden mussten. Die Aufgaben waren aufeinander aufbauend und besaßen einen steigenden Schwierigkeitsgrad. Insgesamt wurde dadurch von den Probanden sukzessive ein Teetimer aufgebaut. Ihnen wurde dazu ein leeres HMISL Modell vorgegeben, welches um die notwendige Verhaltenslogik erweitert werden sollte. Das zugehörige GUI wurde bereitgestellt und ebenso eine Java-Applikation zur Bedienung, die eine Kreuzwippe mit Mittenbetätigung zeigt.

Die Aufgaben waren in mehrere Teilaufgaben unterteilt, die vor Erreichen festgelegter Zeitschranken gelöst werden mussten. Bei den ersten beiden Aufgaben bestand die erste Teilaufgabe jeweils aus einer Analysephase, in der die Probanden die notwendigen Arbeitsschritte zur Erfüllung der Aufgabe mit den Sprachkonzepten der HMISL identifizieren sollten. In den darauffolgenden praktischen Teilaufgaben sollten sie diese Arbeitsschritte nacheinander umsetzen. Um dafür gleiche Ausgangsbedingungen zu schaffen, wurde ihnen nach der ersten Teilaufgabe deren Musterlösung mitgeteilt.

Bei den beiden letzten Aufgaben wurde keine separate Analysephase durchgeführt und auch keine Einteilung in einzelne Teilaufgaben vorgenommen. Stattdessen konnte die Vorgehensweise zur Lösung der Aufgabenstellung frei gewählt werden.

Während der Aufgabenbearbeitung wurde der Erfolg der Probanden sowohl vom Versuchsleiter als auch vom Protokollführer anhand der folgenden Skala bewertet:

- *Keine Probleme:* Aufgabe wurde innerhalb der vorgegebenen Zeit im ersten Versuch selbstständig korrekt gelöst.
- *Lösen mit Trial und Error:* Aufgabe wurde innerhalb der vorgegebenen Zeit selbstständig und korrekt gelöst, es waren jedoch mehrere Versuche notwendig.
- *Nur mit Hilfestellung lösbar:* Aufgabe wurde innerhalb der vorgegebenen Zeit korrekt gelöst, es wurden aber einfache, geschlossene Rückfragen gestellt.
- *Abbruch nach Zeitschranke:* Aufgabe wurde nicht innerhalb der vorgegebenen Zeit gelöst.
- *Falsche Lösung:* Aufgabenbearbeitung wurde innerhalb der vorgegebenen Zeit beendet, es wurde jedoch eine falsche Lösung entwickelt.

Zur Erreichung der in den Akzeptanzkriterien zu Verständlichkeit und Erlernbarkeit festgelegten Schwelle von 80% gilt hierbei sowohl das Lösen mit Trial und Error als auch mit Hilfestellung als akzeptabel, weil es bei einem realen Einsatz üblich erscheint, dass Anfänger erfahrene Entwickler mit einfachen Fragen konsultieren können. Alle bei der Aufgabenbearbeitung aufgetretenen Fehler und alle Rückfragen wurden notiert. Nach Abschluss jeder Aufgabe wurden die Probanden gebeten, mit Hilfe eines Fragebogens die Vorgehensweise in der HMISL zu bewerten. Neben der Bewertung der absoluten Tauglichkeit sollte dabei auch ein Vergleich der Effizienz und weiterer Kriterien mit der bisherigen Vorgehensweise vorgenommen werden.

4.3 Ergebnisse

Für die Untersuchungen wurden insgesamt 11 Probanden ausgewählt, von denen 10 tatsächlich teilnahmen. Der Zentralwert ihrer Erfahrung in der Softwareentwicklung in Vollzeit beträgt 4 Jahre und derjenige ihrer Tätigkeitsdauer im derzeitigen Umfeld 2,8 Jahre. Ein Großteil hatte den Schwerpunkt ihrer Ausbildung in der Informatik absolviert. Als Tätigkeitschwerpunkt innerhalb ihres Teams gab die Hälfte Implementierung an, jeweils ein Proband bezeichnete seine primäre Aufgabe als Werkzeugentwicklung, Architekturentwurf oder Integration, zwei Probanden nannten Architekturentwurf. Der überwiegende Teil der Probanden gab an, keine Erfahrung mit Codegeneratoren und DSLs und nur wenig Erfahrung mit MDSD zu haben (Abbildung 1)[2].

Nach der Erfassung ihres sozio-demografischen Hintergrunds wurden die Probanden um eine Bewertung verschiedener Aussagen zur Nützlichkeit von MDSD im Allgemeinen, für Automotive-HMIs und für ihre derzeitige Tätigkeit gebeten (Abbildung 2).

[2] In Klammern ist der Anteil der Probanden angegeben, welche die entsprechende Frage beantwortet haben.

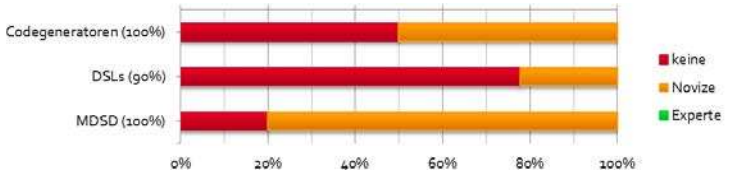

Abbildung 1 – Selbsteinschätzung der Probanden

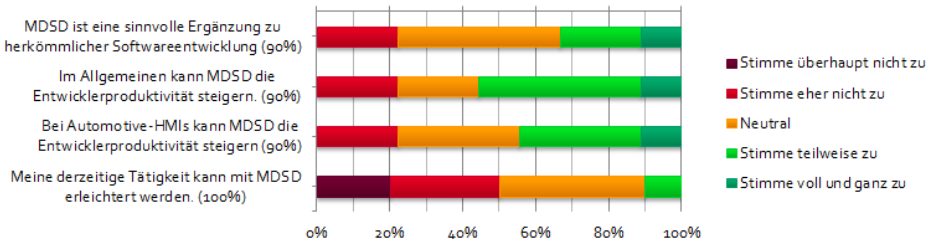

Abbildung 2 – Subjektive Einschätzung vor Kennenlernen der HMISL

Anschließend erhielten sie eine Einführung in die HMISL und sollten dann damit vorgegebene Aufgaben lösen. Dabei wurden sie vom Versuchsleiter und Protokollführer beobachtet und ihr Erfolg unabhängig voneinander bewertet (Abbildung 3).

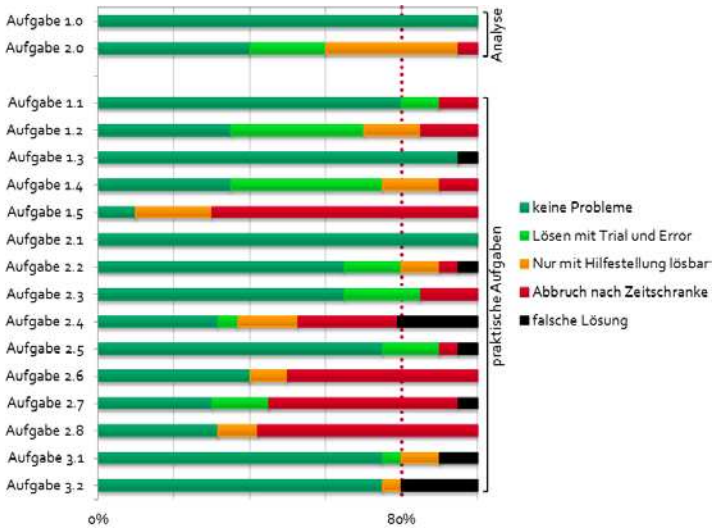

Abbildung 3 – Fremdeinschätzung des Erfolgs der Probanden

Am Ende jeder Aufgabe wurden die Probanden um eine subjektive Bewertung der Vorgehensweise anhand der Verankerung im Fragebogen gebeten[3] (Abbildung 4).

[3] Aufgrund der mehrfachen Befragung und weil Enthaltungen zulässig waren, wurden so von jedem Probanden bis zu drei Bewertungen abgegeben. Dadurch ergeben sich unterschiedliche Beteiligungsquoten und Bewertungsanteile.

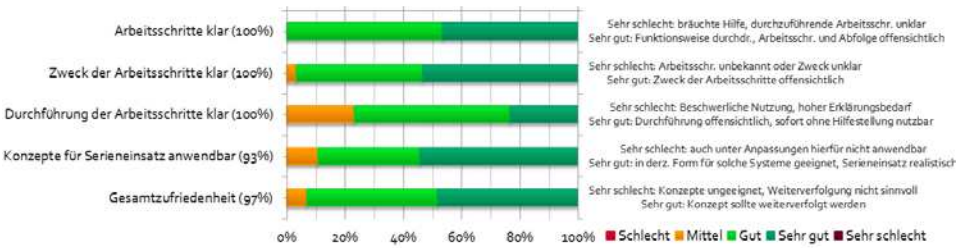

Abbildung 4 – Zusammenfassende Bewertung

4.4 Validität

Obwohl größtmöglicher Wert auf Reproduzierbarkeit gelegt wurde, sind Einflüsse auf die Validität der Ergebnisse nicht auszuschließen. Eine größere Probandenanzahl hätte die Verlässlichkeit der Aussagen verbessert. Aufgrund der begrenzten Zeit für die Untersuchungen konnten zudem mit den Aufgabenstellungen nicht alle Sprachelemente der HMISL abgedeckt werden. Daher wurde der Schwerpunkt auf diejenigen für die Realisierung der Verhaltenslogik gelegt, weil sie vermutlich für die meisten potentiellen Nutzer relevant sind. Es wurden dagegen keine Aufgaben gestellt, bei denen Variabilitätskonzepte, Übersetzungen oder Schnittstellenspezifikationen verwendet werden mussten. Da für jeden Proband und Sprachelement nur eine Untersuchung durchgeführt wurde, können keine Lerneffekte berücksichtigt werden.

Obgleich durch den zuvor erstellten Prototyp große Modelle zur Verfügung standen, wurden sie nicht für die Nutzerstudie eingesetzt, weil eine zu große Einarbeitungszeit der Probanden erforderlich gewesen wäre. Zudem hätte der Programmierstil des Erstellers der Modelle die von den Probanden empfundene Verständlichkeit der Sprache beeinflussen können. Weil somit lediglich kleine Modelle verwendet wurden, kann die Skalierbarkeit der Methodik nur abgeschätzt werden.

4.5 Interpretation

Gemäß der Zielsetzung der HMISL wurden im Rahmen der Benutzerstudie primär Probanden befragt, die eine Ausbildung in der Informatik haben. Deren Aufteilung auf Teams und Aufgabenbereiche bildet einen repräsentativen Querschnitt der relevanten Zielgruppe in dem Projekt, aus dem sie requiriert wurden.

Obwohl in diesem Projekt bereits eine modellgetriebene Vorgehensweise angewandt wird, gab ein Großteil der Probanden an, wenig oder gar keine Erfahrung mit MDSD, DSLs und Codegeneratoren zu haben. Dies ist darauf zurückführen, dass dort die Nutzung dieser Techniken bisher wenigen Spezialisten vorbehalten ist. Bei einem Einsatz der HMISL müssen jedoch weit mehr Entwickler mit dieser DSL arbeiten. Es muss somit erreicht werden, dass ihre Verwendung möglichst ähnlich der Vorgehensweise ist, wie sie es von der konventionellen Implementierung gewohnt sind.

Ein wichtiger Aspekt dafür ist es, dass die Werkzeuge in gewohnter Weise bedient werden. Die Werkzeuge der HMISL basieren auf Eclipse und verwenden die dabei üblichen Bedienungskonzepte. Bei der Nutzerstudie wurden daher auch keine Probleme bei der Benutzung dieser Werkzeuge beobachtet. Es ist darüber hinaus anzunehmen, dass die textuelle Syntax der HMISL ebenfalls dazu beigetragen hat, da sie an Programmiersprachen erinnert, die den Probanden bekannt sind.

Bei der Einschätzung der Probanden vor der Untersuchung hat sich gezeigt, dass eine hohe Skepsis gegenüber den Techniken MDSD, Codegeneratoren und DSLs vorherrscht. Während der Aussage zur Nützlichkeit modellgetriebener Vorgehensweisen im Allgemeinen tendenziell eher zugestimmt wurde, wurde die Nützlichkeit für ihre spezielle Tätigkeit zumeist bezweifelt. Weil jedoch die Bewertungen der Probanden am Ende der Untersuchung deutlich positiver waren, kann angenommen werden, dass sie zunächst noch keine genaue Vorstellung davon hatten, wie sie die Konzepte in der Praxis einsetzen könnten.

Bei der Durchführung der analytischen Aufgaben wurden keine wesentlichen Probleme beobachtet. Man kann daher davon ausgehen, dass die Programmierparadigmen von der Zielgruppe erlernt und verstanden worden sind. Bei den praktischen Aufgaben ergibt sich ein differenzierteres Bild. Unterhalb der Akzeptanzschwelle von 80% liegen hierbei die Ergebnisse für die Aufgaben 1.5, 2.4 sowie 2.6 bis 2.8. Durch eine detaillierte Analyse der dabei aufgetretenen Fehler ließen sich mehrere Ursachen und entsprechende Überarbeitungsvorschläge an der Syntax der HMISL identifizieren:

- *Ungewohnte Operatoren:* In der Aufgabe 1.5 mussten String-Operationen durchgeführt werden. Hierbei haben 70% der Probanden als Konkatenations-Operator für Strings zunächst das Zeichen „+“anstatt „..“ verwendet. Dieser Erwartungshaltung sollte durch Anpassung des Operators entsprochen werden.
- *Asymmetrien:* 50% haben kritisiert, dass das Schlüsselwort „then“ bei Reaktionen verwendet werden muss, nicht aber bei Bedingungen. Zukünftig sollte man dieses bei Reaktionen nicht mehr angegeben werden.
- *Deklaration von Timern:* Bei den Aufgaben 2.4 und 2.6 bis 2.8 mussten Timer eingesetzt werden. Den Probanden war hierbei unklar, dass mit der Formulierung der Reaktion auf einen abgelaufenen Timer dieser auch gleichzeitig deklariert wird. Dies sollte zukünftig getrennt werden, um so dadurch eine Symmetrie zum Vorgehen bei Events und Variablen herzustellen.

Bei der abschließenden Bewertung wurde die Akzeptanzschwelle von 80% für die Tauglichkeit erreicht. Es lag eine hohe Gesamtzufriedenheit mit dem Konzept vor.

Alle Probanden gaben an, dass ihnen die Arbeitsschritte klar waren. Somit bestätigte sich der bereits durch die Fremdeinschätzung der analytischen Aufgaben gewonnenen Eindruck, dass ihnen die Programmierparadigmen keine Schwierigkeiten bereiteten. Die vergleichsweise schlechtere Bewertung der Durchführung der Arbeitsschritte bestätigt wiederum die bei der Fremdeinschätzung der praktischen Aufgaben gewonnene Erkenntnis, dass teilweise Verbesserungsbedarf an der Syntax besteht.

5 Fazit

Die Evaluation der HMISL erfolgte durch Untersuchung von vier Forschungsfragen zu Tauglichkeit, Effizienzgewinnen, Erlernbarkeit und Verständlichkeit.

Durch das Prototyping wurden die grundsätzliche Tauglichkeit und die ausreichende Ausdrucksmächtigkeit nachgewiesen, wofür weitere Konstrukte in der Sprache ergänzt werden mussten. Es wurde zudem offensichtlich, dass es Sonderfälle und Spezifika der Zielsysteme auch zukünftig erforderlich machen werden, individuelle Anpassungen und Erweiterungen an der HMISL zu entwickeln. Sie stellt somit keine fertige Lösung dar, die unmittelbar für die Serienentwicklung geeignet ist, sondern ist vielmehr eine Basissprache, an der spezifische Erweiterungen vorgenommen werden müssen.

Bei der Nutzerstudie wurden keine Probleme mit den grundsätzlichen Programmierparadigmen und den Werkzeugen festgestellt. Insgesamt hat die Mehrheit der Probanden die Vorgehensweise als gut oder sehr gut bewertet und hält die Konzepte für geeignet. Während bei der Mehrheit der praktischen Aufgaben die zu Beginn dieses Kapitels formulierten Akzeptanzkriterien hinsichtlich Erlernbarkeit und Verständlichkeit erfüllt wurden, ist dies bei einzelnen Teilaufgaben nicht der Fall gewesen. Die problematischen Sprachkonstrukte konnten identifiziert werden. Es wurden Anpassungen vorgeschlagen, deren Umsetzung verspricht, bei einer erneuten Durchführung der Nutzerstudie bei allen Teilaufgaben die Akzeptanzkriterien zu erfüllen.

Literaturverzeichnis

[Ger11a]. Gerlach, S.: Neue Kooperationsformen zwischen Fahrzeugherstellern & Zulieferern. JavaSPEKTRUM(5), 2011, S. 8-9.

[Ger11b] Gerlach, S.: The HMISL language for modeling HMI product lines. In (Riener, A., Hrsg.): AutomotiveUI. Universität Salzburg, ICT&S Center, Salzburg, 2011, S. 15-16.

[Ger11c] Gerlach, S.: Ergänzung vorhandener GUI-Modellierungswerkzeuge zur vollständig modellbasierten Entwicklung von Automotive-HMIs. In: (Heiß, Hans-Ulrich et al., Hrsg.): INFORMATIK 2011. GI, Bonn, 2011, S. 293.

[HB88] Herndon, J., Berzins, V.: The Realizable Benefits of a Language Prototyping Language. IEEE Trans. Softw. Eng 14, S. 803-809, 1988.

[KKB96] Kieburtz, R., McKinney, L., Bell, J., Hook, J., Kotov, A., Lewis, J., Oliva, D., Sheard, T., Smith, I., Walton, L.: A software engineering experiment in software component generation. IEEE Computer Society, Washington, DC, USA S. 542-552, 1996.

[Tol05] Tolvanen, J.-P.: Keynote Address II: Domain-Specific Modeling: No One Size Fits All. Springer, S. 279-279, 2005.

[Voe11] Völter, M.: MD*/DSL Best Practices. http://voelter.de/data/pub/DSLBestPractices-2011Update.pdf

[Bet04] Bettin, J.: Prozessauswirkungen von MDSD. OBJEKTspektrum(MDD) 2004.

[SP03] da Silva, P., Paton, N.: User Interface Modeling in UMLi. IEEE Software 20, S. 62-69, 2003.