Entwicklung und Evaluierung einer domänenspezifischen Sprache für SPS-Schrittketten

Bastian Cramer Dennis Klassen Uwe Kastens
University of Paderborn
Department of Computer Science
Fürstenallee 11, 33102 Paderborn, Germany
{bcramer,dennis.klassen,uwe}@uni-paderborn.de

Abstract: Domänenspezifische Sprachen mit passenden Entwurfs- und Transformationswerkzeugen unterstützen Anwender in speziellen Gebieten ihre Entwürfe in Implementierungen umzusetzen. Sind solche Sprachen visuell, so können auch graphische Notationen aus dem Anwendungsgebiet übernommen werden, um die Akzeptanz der Sprache zu verbessern. In diesem Artikel berichten wir über den Entwurf, die Implementierung und Evaluation einer visuellen Sprache, die im industriellen Umfeld zur Steuerung von Robotern einer Produktionsstrecke eingesetzt wird. Der Einsatz eines Werkzeugsystems zur Sprachimplementierung (DEViL [Sch06a]) ermöglichte, dass mit akzeptablem Aufwand Prototypen generiert und Sprachvarianten erprobt wurden.

1 Einleitung

Domänenspezifische Sprachen eignen sich sehr gut, um Entwürfe in speziellen Anwendungsgebieten treffend zu formulieren und mit passenden Werkzeugen in Programme oder andere formale Strukturen zu transformieren. Visuelle Sprachen sind für solche Zwecke besonders nützlich, da Strukturen und Zusammenhänge sehr anschaulich dargestellt werden können. Außerdem können sie gut an etablierte Notationen des Anwendungsgebietes angepasst werden und verkleinern dann die Distanz zwischen den Vorstellungen des Anwenders und den Sprachelementen.

In diesem Artikel berichten wir über eine visuelle Sprache, die wir für die Firma Robert Bosch GmbH entwickelt haben, um damit die Steuerung von Robotern zu formulieren, die Elektromotoren montieren. Mit Entwürfen in dieser Sprache sollen Entwickler, die eine Motoren-Produktionsstrecke konzipieren, ihre Vorstellungen zum Ablauf der Montage an Programmierer vermitteln, welche die Steuerung der Roboter implementieren. Die Randbedingungen für die Entwicklung der Sprache werden in Abschnitt 2 dargestellt.

Die von uns entwickelte Sprache soll eine informelle Notation ablösen, die bisher für diesen Zweck eingesetzt wurde. Damit soll erreicht werden, dass Entwürfe treffender beschrieben und Entwicklungszyklen abgekürzt werden. Der Entwurf der Sprache musste Konzepte der bisher verwendeten Notation gut aufnehmen und Erweiterungen und Präzisierungen sensibel einfügen, damit die Sprache von beiden Entwicklergruppen akzeptiert wurde. Im Abschnitt 3 stellen wir Sprachkonstrukte exemplarisch vor und berichten über die Evaluation der Sprache als akzeptanzsteigernde Maßnahme.

Um mit einer domänenspezifischen Sprache praktischen Nutzen zu erzielen, werden sprachspezifische Werkzeuge benötigt. Wir haben für diese Sprache u.a. einen visuellen Strukturediteor, Übersetzer für SPS nach XML und einen Codegenerator für SPS implementiert und in den Entwicklungsprozess für die Produktionsstrecken eingebettet. Abschnitt 4 vermittelt einen Eindruck dieser Aufgaben jenseits der Sprachentwicklung im engeren Sinne.

Neben den grundsätzlichen Aufgaben zur Implementierung von Sprachen erfordert eine visuelle Sprache in erheblichem Maße zusätzliches technisches und konzeptionelles Wissen sowie weiteren Entwicklungsaufwand. Wenn man trotzdem noch Entwurfsvarianten erproben will, um die Akzeptanz zu verbessern, ist der Aufwand für eine manuelle Implementierung nicht akzeptabel. Wir haben für die Realisierung dieser Sprache das von uns entwickelte Werkzeugsystem DEViL [Sch06a] eingesetzt. Es generiert vollständige Sprachimplementierungen mit visuellen Strukturediteoren als Frontend aus Spezifikationen hohen Abstraktionsniveaus [Sch06b]. Im Abschnitt 5 vermitteln wir einen Eindruck von dem Einsatz und der Wirksamkeit des Systems.

Einige Hinweise auf verwandte Arbeiten und eine Zusammenfassung beschließen den Artikel.

2 Vorstellung der Domäne

Die Firma Robert Bosch GmbH produziert an ihrem Standort in Bühl kleine Elektromotoren in großen Stückzahlen für vielfältige Einsätze in Kraftfahrzeugen. Auf langen Produktionsstrecken mit vielen einzelnen Stationen bauen Roboter die Motoren schrittweise zusammen und prüfen sie. Für jeden neuen Motortyp muss die Produktionsstrecke neu eingerichtet werden. Dazu wird für jede Station der Aufbau des Roboters, die Zuführung des Materials und die Weiterleitung der Produkte entworfen. Darauf abgestimmt werden die Arbeitsschritte des Roboters geplant und in speicherprogrammierbaren Steuerungen (SPS [Aue89]) programmiert.

Bei der Entwicklung der motorspezifischen Produktionsstrecke kooperieren zwei Gruppen von Entwicklern mit unterschiedlichen Aufgaben: Die hier „Projektleiter“ genannte Gruppe entwirft den mechanischen Aufbau jeder Station und den Einsatz der Roboter, Greifarme, Schub- und Dreheinrichtungen, Taster und Sensoren. Die zweite Gruppe, hier als „Programmierer“ bezeichnet, entwickelt Programme zur Steuerung der Komponenten mit speicherprogrammierbaren Steuerungen. Die beiden Entwicklergruppen kommunizieren über den geplanten Ablauf der Produktionsschritte der Stationen. In mehreren Iterationen werden die Pläne verfeinert, in Programme umgesetzt, an den Geräten getestet und verbessert. Die Pläne der Abläufe, sogenannte „Schrittketten“ werden in einer speziellen Notation beschrieben, zwischen den Entwicklergruppen ausgetauscht, fortgeschrieben und dokumentiert. Die Notation enthält graphische Elemente zur Skizzierung von Ablaufschritten und textuelle Annotationen für technische Angaben. Abbildung 2 und 3 zeigen Beispiele dazu.

Der Entwicklungsprozess einer Produktionsstrecke wird durch die Projektleiter mit der Entwicklung der mechanischen Konstruktion begonnen (Abbildung 1, Position 1). Aus den dabei entstehenden Dokumenten wird mit einem Generator ein Grundprojekt erstellt

(Abbildung 1, Position 2). Dabei wird für jedes aktive Element der Produktionsstrecke eine sogenannte Schrittkette festgelegt, die den Ablauf des Elements beschreiben soll. Die Programmierer extrahieren die einzelnen Schrittketten (Abbildung 1, Position 3). Mit dem Schrittketten-Dokumentationssystem, das in der Abteilung verwendet wird, können die vorhandenen Schrittketten in festgelegter Form als Schrittketten-Ablaufzettel ausgedruckt werden (Abbildung 1, Position 4). Abbildung 3 stellt beispielhaft Bearbeitungszustände eines Ablaufzettels dar und soll einen Eindruck über die bei Bosch verwendeten Dokumente zur Beschreibung der Schrittketten verschaffen. Eine genauere Ansicht für die Schritte und Konditionen ist in der Abbildung 2 zu finden. Diese Vorlagen werden durch Projekleiter, gemäß den geplanten Bewegungsabläufe, händisch bearbeitet (Abbildung 3) und an die Programmierer übergeben (Abbildung 1, Position 5). Programmierer setzen die Modelle in SPS-Code um und können daraus SPS-Programme erzeugen oder aktuelle Ablaufzettel zur Kontrolle erneut ausdrucken, bis der gewünschte Ablauf der Maschine erreicht wurde und das SPS-Program vollständig ist.

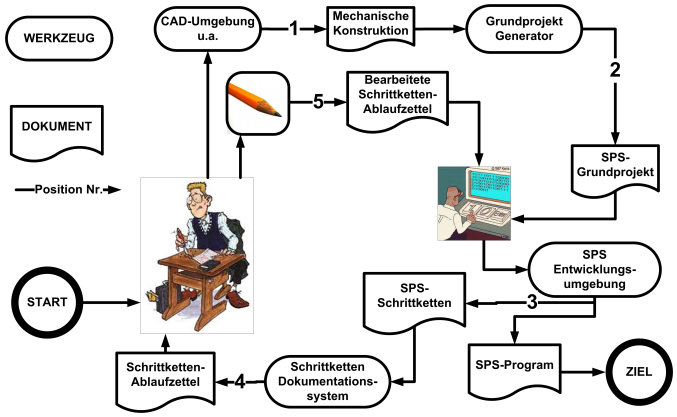


Abbildung 1: Entwicklungsprozess vor der Einführung der DSL

In der Abbildung 2 sind drei Schritte genauer dargestellt. Als Beispiel betrachten wir den Schritt N16, in dem die Aktion „Parkpos.anfahren“ durchgeführt wird. Das grau unterlegte Kästchen stellt den Schritt dar. Darüber sind zwei vertikale Linien angeordnet, diese repräsentieren eine ODER-Verknüpfung zwischen drei Konditionen, unter denen dieser Schritt geschaltet wird. Die untereinander aufgelisteten Variablennamen sind UND-verknüpft. Unter dem Schritt können mehrere Aktionen aufgeführt sein, die in dem Schritt durchzuführen sind, bevor die Konditionen für den nächsten Schritt geprüft werden. Eine Schrittkette kann über hundert Schritte besitzen und erstreckt sich dann über mehrere Seiten.

Die Schrittketten-Ablaufzettel sind während des gesamten Entwicklungsprozesses im Umlauf (Abbildung 1, Punkt 7,9). Alle Änderungen an dem Ablauf oder den Maschinen fordern stets den Einsatz des Schrittketten-Ablaufzettels. Häufig sind die Änderungen nur gering, wie z.B. das Entfallen einiger Schritte, die Änderung der Schrittreihenfolge oder eine einfache Namensänderung der Schritte. Die Abbildung 3 zeigt einen Auszug aus einem laufenden Projekt mit durchgeführten Korrekturen.

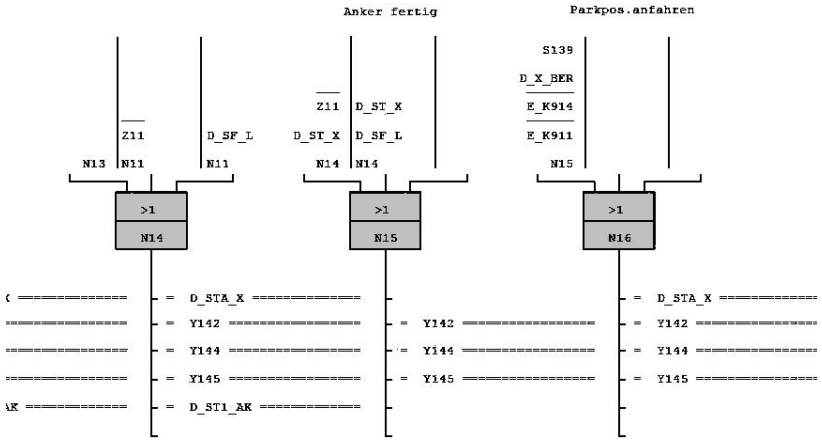


Abbildung 2: Schritte, dargestellt mit dem Schrittketten-Dokumentationssystem

Die größten Engpässe sind die Kommunikation zwischen den Entwicklergruppen und die dabei entstehenden Dokumente. Die Beschreibungsmöglichkeiten sind in dieser Notation sehr begrenzt und passen nicht mehr zum aktuellen SPS-Standard. Das führt immer wieder zu ungenauen oder missverständlichen Beschreibungen und erfordert zusätzliche Iterationen (Abbildung 1, Punkt 4-9), bis das erstellte SPS-Programm dem gewünschten Bewegungsablauf entspricht.

Diese Situation sollte im Rahmen des hier beschriebenen Projektes durch eine domänenspezifische Sprache verbessert werden.

3 Sprachentwurf

Wie aus dem vorhergehenden Abschnitt hervorgeht, wird eine domänenspezifische Sprache benötigt, welche den Aufgaben angemessen ist. In diesem Abschnitt wird die Vorgehensweise für einen solchen Sprachentwurf vorgestellt. Die SPS-Programmiersprachen spielen dabei eine wesentliche Rolle, da die DSL Programme für SPS beschreiben soll. Dafür wird zu Beginn eine Übersicht über die SPS-Sprachen gegeben. Anschließend wird der Sprachentwurf und die dazu verwendeten Elemente aus der vorhandenen Notation beschrieben. Im Anschluss daran werden die Evaluierung und die daraus resultierende DSL beschrieben.

3.1 SPS-Konzepte als Grundlage der DSL

Die DSL soll speziell für die Entwicklung der Schrittketten eingesetzt werden, wofür nur bestimmte Teile der SPS-Sprachen verwendet werden. Der Standard IEC 61131 [Inf03] vereinigt fünf Sprachen für SPS. Die Ablaufsprache (AS) dient der Gliederung und Orga-

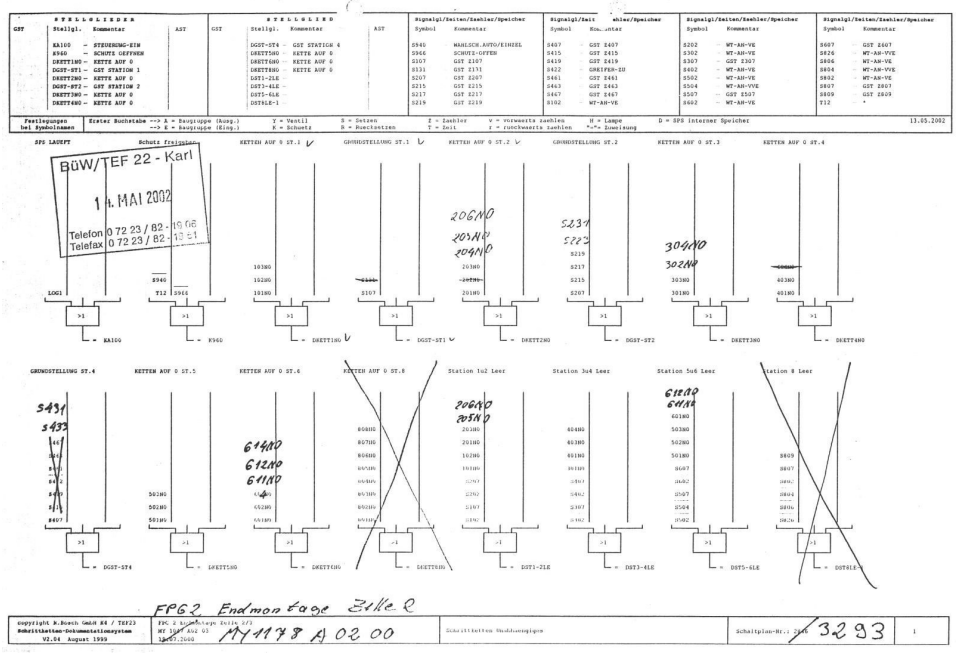


Abbildung 3: Schrittketten-Ablaufzettel mit Änderungen

nisation einer Schrittkette und bildet einen gerichteten Graph aus einer Menge von Schritten und Transitionen (Abbildung 2). Die Schrittaktionen und die Transitionsbedingungen können in den weiteren SPS-Sprachen beschrieben werden. Zu ihnen gehören zwei textuelle Sprachen (Tabelle 1), Anweisungsliste (AWL), Strukturierter Text (ST), und zwei Sprachen mit visuellen Elementen (Tabelle 1), Kontaktplan (KOP), Funktionsbaustein (FBS).

Die AWL fasst mehrere Anweisungen zusammen. Operanden können Werte zugeordnet werden, z.B. bei der Deklaration von Variablen. Die ST besteht aus Ausdrücken, die einen Wert liefern. Dazu gehören z.B. mathematische Ausdrücke oder boolesche Funktionen. Außerdem ist das Beschreiben verschiedener Konstrukte wie Auswahl- oder Wiederholungsanweisungen möglich.

Mit den Sprachen KOP und FBS wird ein Netzwerk in Form von Linien und Blöcken dargestellt. Diese Notation ist den früher benutzten Schaltplänen aus der Elektrotechnik nachempfunden und beschreibt Stromfluss-Pläne.

FBS stellt eine Menge von Funktionsbausteinen dar. Durch Verknüpfung mehrerer Funktionsbausteine miteinander können komplexe Funktionen realisiert werden. Dazu gehören diverse mathematische Funktionen oder vollständige Motorsteuerungen.

Einen Eindruck der Sprachen vermittelt Tabelle 1. In allen vier Beispielen wird eine UND-Funktion mit Eingängen % X2,5 und % X2,3 und dem Ausgang TRANS78 dargestellt.

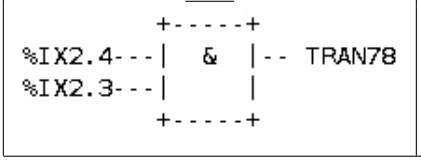
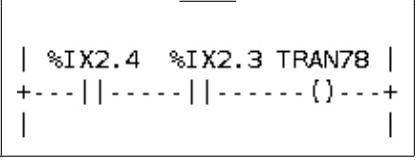
ST	AWL
:= %IX2.4 & %IX2.3	LD %IX2.4 AND %IX2.3
FBS	**KOP**
	

Tabelle 1: SPS-Sprachen für verschiedene Zwecke: ST, AWL, FBS und KOP

Mit dem Expertenwissen über SPS kann mit der Entwicklung der domänenspezifischen Sprache begonnen werden. Dazu wird ermittelt, was genau die neue DSL beschreiben soll. Im Vorfeld dieser Arbeit ist eine Skizze (Abbildung 4) entstanden, welche die grundlegenden Strukturen zeigt und die Darstellungswünsche der Benutzer aufgenommen hat. Daraus ist ersichtlich, dass die neue Sprache alle Sprachelemente einer Schrittkette in einer grafischen Übersicht darstellen sollte. Ein weiterer Anhaltspunkt für die visuelle Sprache sind die Schrittketten-Ablaufzettel (Abbildung 3).

Bereits an dieser Stelle sind Gemeinsamkeiten und Gegensätze zwischen vorhandenen Werkzeugen und den Wünschen der Entwickler erkennbar. Z.B. sollte der Fluss der Schrittketten von links nach rechts beibehalten werden, entsprechend der bereits vorhandenen Notation in den Schrittkettenablaufzetteln (Abbildung 3, 4). Geändert werden sollte die Beschreibung der Transitionen und Bedingungen: Auf den Schrittkettenablaufzetteln (Abbildung 3) werden Bedingungen modelliert, unter welchen der dazugehörige Schritt geschaltet wird; in der neuen Sprache (Abbildung 4) werden Transitionen modelliert, die zum Verlassen des Schrittes führen.

Als erstes werden die zu modellierenden Elemente der SPS-Sprachen ermittelt. Dafür wird das Expertenwissen aus dem Umfeld benötigt, unterstützend dazu werden die vorhandenen Projektdokumente untersucht. Sobald der Satz der zu modellierenden Elemente feststeht, kann dieser in einer abstrakten Struktur für DEViL [Sch06a] implementiert werden. Aus dieser abstrakten Struktur ist DEViL bereits in der Lage einen Strukturedитor mit einer simplen Baumdarstellung zu generieren. Bereits hier sind mit den generierten Strukturedитoren Evaluierungstests möglich. Dabei kann untersucht werden, ob sich bereits vorhandene Projekte mit den Strukturbäumen ausdrücken lassen. Nachdem die abstrakte Struktur feststeht, kann mit der Modellierung der visuellen Komponenten der DSL begonnen werden.

3.2 Evaluierung der DSL

In diesem Projekt haben wir der Evaluierung besondere Beachtung geschenkt. Dazu wurden verschiedene Methoden und Vorgehensweisen angewendet, mit denen unterschiedliche Effekte evaluiert wurden, z.B. Kommunikation zwischen den Benutzern, Verständlichkeit der Modellierungssprache usw.. Die dabei erzielten Ergebnisse sind z.T. benutzer-

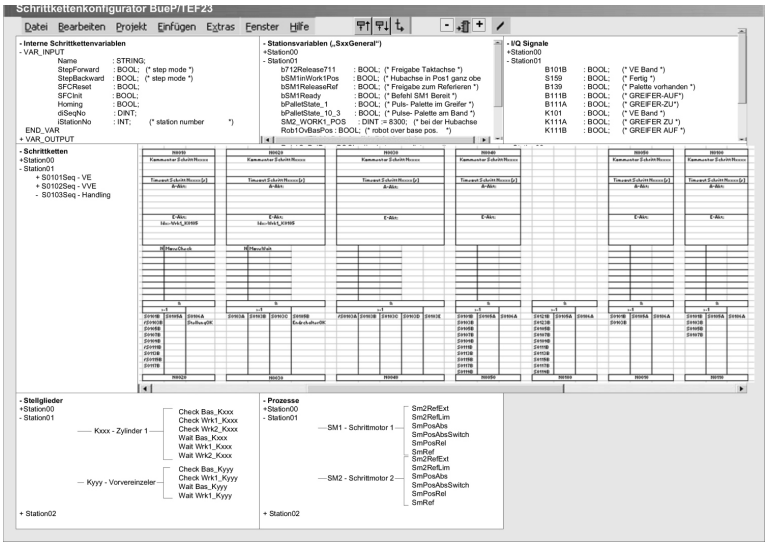


Abbildung 4: Erste Skizze des Schrittkettenkonfigurators

abhängig und müssen richtig interpretiert werden. Außerdem sollte beachtet werden, dass sich die Wünsche und Meinungen der Entwickler im Verlauf der Evaluierung ändern, so ist z.B. in der Abbildung 4 zu sehen, dass die Benutzer alle Elemente zum Konfigurieren einer Schrittkette in einer einzigen Sicht dargestellt wünschen. Dieser Wunsch hat sich im Verlauf der Evaluierung geändert, da realistische SPS-Projekte in einer Sicht umfangreich und zu unübersichtlich werden.

Neben dem Expertenwissen und Kenntnissen über die Domäne muss für die Evaluierung der DSL der Kenntnisstand der Benutzer ermittelt werden. Für diesen Zweck sind Interviews und Beobachtungen vor Ort notwendig. Die Ergebnisse können für die Dialogmodellierung verwendet werden, die beschreibt, wie sich das System zu verhalten hat und welchen Arbeitsfluss die Benutzer beim Erledigen ihrer Aufgaben haben. Außerdem kann mit Hilfe der Interviews herausgearbeitet werden, wie die Benutzungsschnittstelle auszusehen hat. Der Aufbau einer Grundakzeptanz ist für die weitere Entwicklung enorm wichtig. Erst wenn diese erreicht wurde, kann mit der Evaluierung der DSL fortgefahren werden.

Die Evaluierung wurde parallel zum Prototyping und der Implementierung durchgeführt. Bereits eine Skizze auf Papier (Abbildung 4) verkörpert den ersten Prototyp und die ersten Schritte der Evaluierung. Das “Rapid Prototyping” wurde durch den Einsatz des Generators DEViL ermöglicht. Da sich sowohl einzelne Teile der Benutzungsschnittstelle als auch Sprachelemente separiert evaluieren lassen, kann dadurch den Benutzern eine Vielzahl von Alternativen angeboten werden. Ein vielfältiges Angebot ist notwendig, da die Benutzer nicht den Überblick über die verschiedenen Modellierungstechniken besitzen. Das bedeutet, die Benutzer können nicht direkt mitteilen, welche Modellierungsmethode sie bevorzugen, sondern können aus den präsentierten Methoden eine Auswahl treffen und

diese bewerten.

Die Auswahl der Benutzer für die Evaluierung beschränkt sich auf die Mitarbeiter in der TEF-Abteilung bei Bosch in Bühl. Es ist wichtig, dass beide Entwicklergruppen, Programmierer und Projektleiter, bei der Evaluation vertreten sind. Denn sie haben unterschiedlich tiefe Kenntnisse von SPS-Konstrukten, welche die DSL stark prägen.

- Die Programmierer sind im Umgang mit SPS erfahrener, das sie bereits SPS-Software entwickelt und mit grafischen Editoren gearbeitet haben.
- Die Projektleiter programmieren nicht selbst. Ihr Bezug zu der SPS entsteht über den mechanischen Ablauf der Maschinen und die Schrittketten-Ablaufzettel.

Aus den Methoden, die für solche Evaluierungen einschlägig sind [SCK07], haben wir in diesem Projekt folgende eingesetzt:

Interview In einem Interview werden Benutzer aus verschiedenen Entwicklergruppen einzeln interviewt. Das Gespräch dient dazu, die Meinungen und die Aussagen der Benutzer zu dem System zu sammeln und zu untersuchen. Vorteilhaft ist, dass auf die Benutzer einzeln eingegangen werden kann und somit unvorhergesehene Fragen und Beanstandungen geklärt werden können.

Kontrolliertes Experiment Bei einem kontrollierten Experiment wird den ausgewählten Benutzern eine Aufgabe gestellt, die sie mit dem System bewältigen müssen. Dabei wird beobachtet, welche Entwicklergruppen in welchen Bereichen Schwierigkeiten haben.

Feld-Beobachtung Eine der wichtigsten Evaluierungsmethoden für diese Arbeit ist der Feldtest. Dabei werden die Benutzer während der Arbeit mit dem neuen System beobachtet. Im Vergleich zum kontrollierten Experiment ist diese Methode praxisrelevanter und kann unvorhergesehene Ergebnisse liefern.

Als eine wirksame Evaluierungsmethode hat sich eine Mischung aus Gruppeninterview, Feld-Beobachtung und kontrolliertem Experiment herausgestellt. Dabei wurden mehrere Benutzer aus beiden Gruppen zu einer Präsentation eingeladen. Der Ablauf der Versuche wurde dann wie folgt durchgeführt: Eine Schrittkette sollte zwischen mehreren Benutzern diskutiert und gegebenenfalls editiert werden. Die Änderungen an der Schrittkette, die im Verlauf des Tests entstanden, wurden z.B. von einem Projektleiter durchgeführt. Dabei konnten die Testpersonen ihre persönliche Meinung einbringen. Durch solche Experimente konnten die Schwierigkeiten in der Benutzung der Prototypen des Strukturediteors herausgestellt und beseitigt werden. Die eindeutigsten und aussagekräftigsten Beobachtungen konnten bei der Kommunikation zwischen den Entwicklergruppen gemacht werden. Durch diese Experimente konnte die Darstellung der Benutzungsschnittstelle und der Softwarebeschreibungssprache an die Kommunikation zwischen den Entwicklergruppen angepasst werden. Die unten aufgeführten Evaluierungstests wurden sowohl mit einzelnen Personen, als auch, wie hier beschrieben, in einer Gruppe abwechselnd durchgeführt.

Der erste Test sollte die Aufteilung der visuellen Sichten in den verschiedenen Prototypen herausstellen. Dabei galt es herauszufinden, wie gut sich die Benutzer in dem Strukturediteor zurecht finden und welche Informationen in einer Sicht gebündelt werden oder eine eigene Sicht erhalten sollten. Dazu wurden verschiedene Prototypen des Strukturediteors

bereitgestellt, in denen Ausschnitte aus den SPS-Programmen, formuliert in der experimentellen DSL, in gemischten Gruppen aus Programmierern und Projektleitern diskutiert wurden.

Die Ergebnisse dieser Tests wurden fortlaufend analysiert, um neue Prototypen bereitzustellen. Durch eine große Prototypenanzahl konnte eine für die Benutzer optimale Benutzungsschnittstelle geschaffen werden, die sich recht deutlich von den Anfangswünschen der Benutzer (Abbildung 4) unterscheidet. Dabei entstanden fünf Sichten, in die sich der Strukturedитor aufteilt: Hauptsicht, Schrittketten, Schrittketten-Variablen, Schrittketten- Aktionen und globale Variablen.

Der zweite Test bezog sich auf die Darstellung der Schritte und der Schrittketten. Das Ziel war, herauszufinden, wie verständlich die Schrittketten und Schritte mit der neuen DSL dargestellt werden. Dabei mussten die Testpersonen die Informationen aus einer vorgegebenen Schrittkette notieren. Zusätzlich konnten die Benutzer in einem Interview die Eindrücke über die Darstellungsart und die Aufteilung der Informationen auf dem Bildschirm mitteilen.

Die Programmierer und die Projektleiter verfolgen unterschiedliche Ziele und haben sehr unterschiedliche Blickwinkel auf die Schrittketten. Daher konnten die aus diesem Test erhaltenen Ergebnisse zwischen den verschiedenen Entwicklergruppen nicht verglichen werden. Die Projektleiter hatten, wie erwartet, anfänglich Schwierigkeiten zwischen den Konditionen und Transitionen zu unterscheiden. Nur eine von fünf Testpersonen aus dieser Gruppe hat die Veränderung und die Bedeutung sofort erkannt. Mit dem gleichen Ergebnis konnten die Schrittaktionen von den Eingangs- und Ausgangsaktionen nicht differenziert werden. Dieses Ergebnis konnte für diese Entwicklergruppe als positiv verbucht werden, da sich die Benutzer auf die für sie relevanten Aufgaben konzentrieren konnten. Aus der Gruppe der Programmierer gab es lediglich drei Testpersonen. Ohne große Mühe wurde die Darstellung von dieser Entwicklergruppe sofort erkannt und vollständig interpretiert.

In einem weiteren Test wurde die Darstellung der visuellen DSL untersucht. Nachdem in vorangegangenen Tests die Bedeutung der verschiedenen Aktionen in einem Schritt und die Bedeutung der Transitionen geklärt wurden, konnten weitere Untersuchungen dieser Elemente durchgeführt werden. Dabei konnte speziell auf die Darstellungen der einzelnen Elemente wie Aktionen in einem Schritt, Transitionen usw. eingegangen werden. Den Testpersonen wurden dann z.B. verschiedene Transitionen vorgegeben, die interpretiert und beschrieben werden mussten. Die Darstellung der Transitionen wurde sowohl von den Projektleitern als auch von den Programmierern als intuitiv und leicht verständlich eingestuft.

Abschließend wurden mehrere Tests des gesamten Werkzeugs durchgeführt. Dazu mussten die Benutzer vorhandene Schrittketten öffnen, bearbeiten und anschließend abspeichern. Die Ergebnisse aus diesen Tests sollten Schwächen des Systems aufzeigen. Nach jedem Durchlauf wurden die Beanstandungen ausgebessert und ein neuer Testdurchlauf gestartet.

Für die Konstrukte der unterschiedlichen SPS-Sprachen wurden durch gezielte Evaluierung geeignete Darstellungen gefunden. Dabei können Schrittketten mit sämtlichen beinhalteten Elementen in einer Ansicht dargestellt werden, sodass ein Benutzer eine Schritt-

kette mit demselben Strukturediteor in derselben Ansicht editieren kann. Die Evaluierung ging so weit, dass z.B. selbst die Darstellung der Kommentare untersucht wurde. Diese sind für die Benutzer beim Erledigen ihrer Aufgaben an einigen Stellen elementar wichtig und an anderen Stellen störend und überflüssig.

Schrittketten, Schritte: Schritte werden in einer Schrittkette nach wie vor in einem Fluss von links nach rechts dargestellt und bestehen aus drei Teilen. Im oberen Bereich, wie in der Abbildung 5 zu sehen, befindet sich der sogenannte Schrittkopf, dem der Benutzer die auszeichnenden Informationen über einen Schritt entnehmen kann. Die dargestellten Informationen sind: Name des Schrittes, Kommentare und Timer-Attribute. Hinweisend werden ganz oben die Schrittnamen der Vorgängerschritte eingeblendet, wie in der Abbildung 5 sichtbar. Weitere Schrittattribute werden in der Ansicht nicht dargestellt, da sie bei dem Entwurf und der Übersicht für die Benutzer zweitrangig sind. Diese können jedoch durch ein zusätzliches Dialogfenster erreicht werden. Im mittleren Teil sind die Aktionen des Schrittes untergebracht, darunter sind die Transitionen aufgeführt.

Schritt-Aktionen In der Abbildung 5 sind einige Aktionen in einem Schritt zu sehen. Auch wenn die Schrittaktionen in AWL anders als die Eingangs- und Ausgangsaktionen in ST sind, wurde die Darstellung annähernd gleich gehalten, um ein einheitliches Bild der logisch zusammen gehörenden Aktionen zu bekommen. Besonders sei auf den Schritt N0025 in der Abbildung 5 hingewiesen. In der Ausgangsaktion dieses Schrittes ist eine bedingte Anweisung dargestellt. Solche Darstellungen sind mit den Schrittketten-Ablaufzetteln nicht möglich.

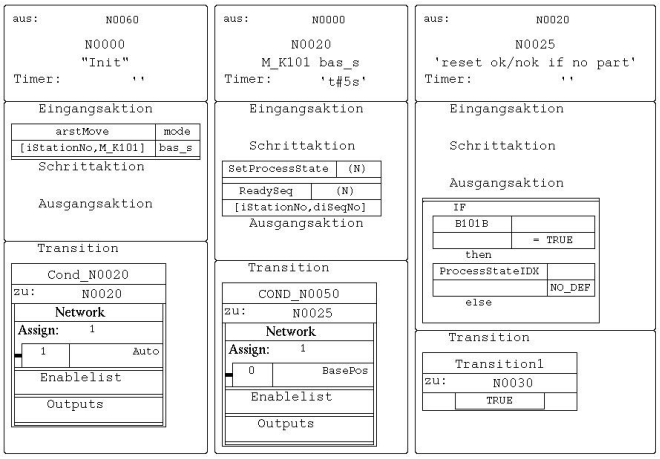


Abbildung 5: Schritte/Aktionen/Transitionen

Transitionen Eine Transition beschreibt die Bedingung und den Übergang zum nächsten Schritt. Wenn ein Schritt mehrere Transitionen besitzt, werden diese nebeneinander aufgereiht. Eine Transition ist, wie der Schritt, in drei Teile gegliedert. Als erstes ist der Name der Transition platziert, gefolgt von dem Zielschritt. Unten werden die Transitionsbedingungen dargestellt.

4 Werkzeugeinbettung

Die neue domänenspezifische Sprache wurde im Rahmen des Projekts in einem visuellen Strukturedi tor (Abbildung 7) realisiert, der mit einem Code Generator ausgestattet wurde. Damit die Eingliederung in die vorhandene Entwicklungsroutine möglich ist, wurde zusätzlich ein Übersetzer für SPS nach XML implementiert. Die Abbildung 6 gibt eine Übersicht des Entwicklungsprozesses mit dem neuen Werkzeugsystem im Vergleich zu dem ursprünglichen Prozess in Abbildung 1. Nach der Einführung des neuen Systems beginnt der Entwicklungsprozess, wie bereits in Abschnitt 2 beschrieben, mit der mechanischen Konstruktion (Abbildung 6, Position 1) der Produktionsstrecke. Sobald die benötigten Schrittketten festgelegt wurden, kann mit ihrer Entwicklung, in dem Schrittkettenkonfigurator, angefangen werden (Abbildung 6, Position 2). In dem Schrittkettenkonfigurator können Benutzer die Schrittketten beliebig mit Schritten, Aktionen und Transitionen vervollständigen (Abbildung 6, Position 3,4).

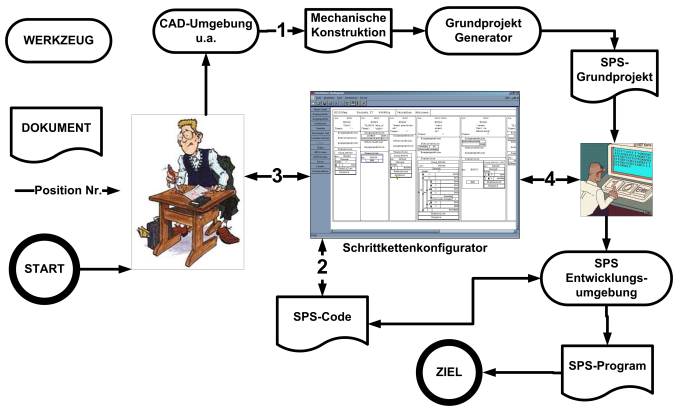


Abbildung 6: Veränderter Entwicklungsprozess mit dem Schrittkettenkonfigurator

Die Projektleiter beschreiben im Wesentlichen, wie viele Schritte benötigt werden und welche Aktionen in diesen Schritten enthalten sein müssen. Die genaue Parametrisierung der Aktionen ist die Aufgabe der Programmierer. Weiterhin beschreiben die Projektleiter einfache Transitionen, die ebenfalls durch Programmierer nachträglich vervollständigt werden. Alle Informationen, die das Wissen der Projektleiter über SPS-Programmierung übersteigen, werden in Form einfacher Kommentare in die betroffenen Module eingetragen. Enormer Vorteil in der Kommunikation ist, dass die Benutzer bei einer gemeinsamen Schnittstelle über die gleichen Elemente und Darstellungen sprechen.

Nachdem ein erster Entwurf durch die Projektleiter erfolgt ist, übernehmen die Programmierer die Beschreibung. Für die Einbettung der Schrittketten in das Grundprojekt, in der Entwicklungsumgebung IndraWorks von Rexroth [Ind], wird aus dem Schrittkettenkonfigurator direkt SPS-Code generiert. Somit kann SPS-Code zwischen dem Schrittkettenkonfigurator und IndraWorks ausgetauscht werden. Der zweite nennenswerte Verbesserungspunkt im Entwicklungsprozess ist, dass die Programmierer die erhaltenen Schrittketten

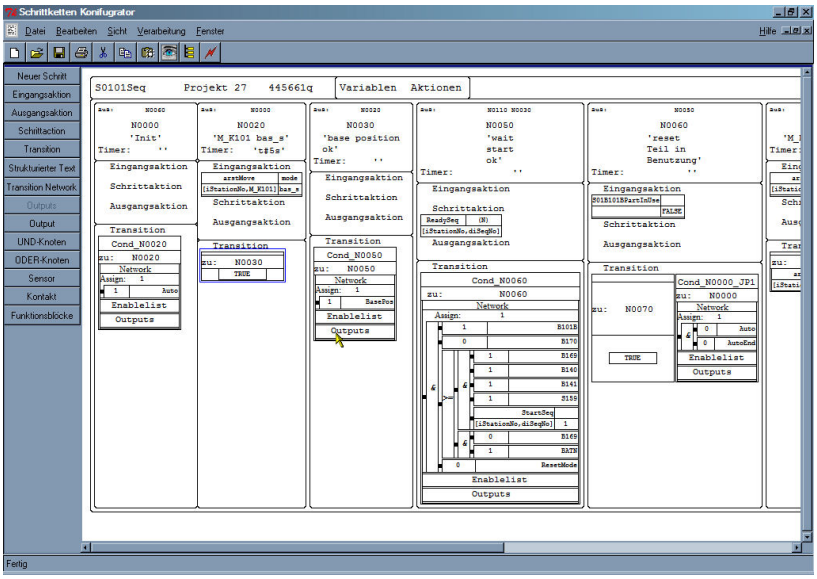


Abbildung 7: Schrittkettenkonfigurator

nicht neu erfassen müssen, sondern dort anknüpfen können, wo die Projektleiter aufgehört haben. Verpackt in einem Strukturedi tor ist die domänenspezifische Sprache perfekt in den Entwicklungsprozess integriert und unterstützt die Entwickler angemessen.

5 Generatorsystem

Für die Entwicklung der visuellen DSL und ihrer Implementierung haben wir den Generator DEViL [Sch06a] eingesetzt. DEViL basiert auf den Konzepten des Übersetzergeneratorsystems Eli [Eli] und ist der Nachfolger von VL-Eli [KS02]. DEViL generiert aus Spezifikationen hohen Abstraktionsniveaus vollständige graphische Strukturedi toren, die aus einer Multifensterumgebung bestehen und alle Eigenschaften heutiger visueller Editoren wie Laden und Speichern von Dokumenten im XML-Format, Kopier- und Einfüge- sowie Undo/Redo-Operationen, Drucken von Dokumenten und Suchen bieten.

Zentrale Datenstruktur der generierten Editoren ist ein abstrakter Strukturbaum. Er wird aus einer Spezifikation generiert, die einem klassenbasierten Entwurf ähnelt. In der Spezifikationssprache sind die Definition von Attributen mit vor- oder selbstdefinierten Typen, Referenzen auf Sprachkonstrukte sowie Aggregation von Sprachelementen und Mehrfachvererbung erlaubt. Aus solch einer Sprachspezifikation kann DEViL bereits einen rudimentären Strukturedi tor mit einer Baumansicht generieren. Dadurch konnten wir schon mit unterschiedlichen Strukturen unserer DSL experimentieren, bevor wir die Visuelle Repräsentation der Sprachelemente entworfen hatten.

Um eine visuelle Sprache zu definieren, benutzt DEViL das Konzept der visuellen Muster.

Visuelle Muster sind in visuellen Sprachen häufig vorkommende Repräsentationen wie Listen, Formulare, Mengen oder Tabellen. Der Sprachentwickler kann aus einer großen Bibliothek von Mustern wählen, diese anpassen und dann einfach auf das Sprachmodell anwenden (Abbildung 8). Alle für die graphische Darstellung der Sprachelemente notwendigen Implementierungen leistet das System dann automatisch. Soo konnten wird mit recht geringem Aufwand die graphischen Darstellungen von Sprachelementen ändern, wenn die Evaluation dazu den Anlass gab. Des Weiteren kann der Sprachentwickler eine Code Erzeugung für den Struktureditor definieren. Hier stehen ihm alle Möglichkeiten des Eli Übersetzergeneratorsystems zur Verfügung, insbesondere die Unparser. Mit diesen Mitteln haben wir die Erzeugung von SPS-Code realisiert. Die von DEViL generierten Struktureditoren erlauben es dem Benutzer, immer syntaktisch korrekte Programme zu erzeugen. Um komplexere semantische Analysen durchzuführen, stehen viele weitere vordefinierte Funktionen bereit und über eine Sprachschnittstelle können C/C++ bzw. Tcl-Funktionen eingebunden werden.

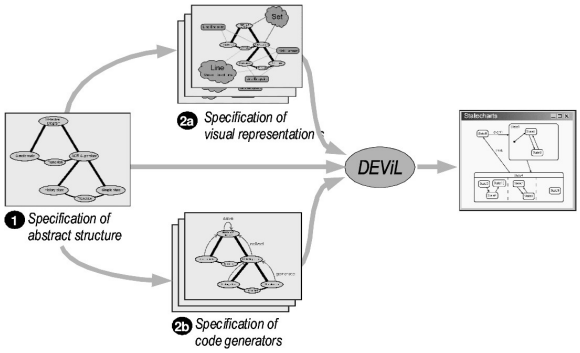


Abbildung 8: Konzept des DEViL-Systems

6 Verwandte Arbeiten

Es existiert eine Vielzahl verschiedener Werkzeugsysteme und Struktureditoren für SPS-Software, die unter anderem auch über visuelle Darstellungen, visuelle Sprachen und Code Generatoren verfügen. Auch gibt es verschiedene Werkzeugsysteme, mit denen Struktureditoren aus Spezifikationen generiert werden. Im Rahmen dieses Papiers können wir lediglich eine kleine Übersicht darüber geben.

Wie schon aus dem SPS-Standard hervorgeht, gibt es visuelle Notationen für SPS und umfangreiche Entwicklungsumgebungen für SPS-Software, wie z.B. IndraWorks [Ind]. Diese Programmierumgebung bietet leider keine Möglichkeit die Darstellungen der Sprachen zu verändern. Da diese Umgebung für professionelle Softwareentwickler ausgerichtet ist, ist sie für unerfahrene Benutzer nicht gut geeignet. Durch einfachere und besser angepasste Notationen können auch die unerfahrenen Benutzergruppen erreicht werden.

Werkzeugsysteme zur Entwicklung visueller Struktureditoren sind z.B. GenGed [Bar98],

MetaEdit+ [Con02] und das in diesem Projekt verwendete DEViL, so wie Eclipse basierende Plattformen wie oAW oder GMF [BSM+03]. Das System GenGed z.B. basiert auf einer als Graph modellierten abstrakten Struktur. In DEViL dagegen werden Sprachkonstrukte durch attributierte Grammatiken spezifiziert und als attributierte Bäume mit Querreferenzen repräsentiert, was eine entscheidende Rolle bei der Strukturmodellierung und den Sicht-Definitionen spielt. Einen ähnlichen Ansatz der Modellierung findet man beim MetaEdit+. Dieses System hat leider eine eingeschränkte Flexibilität bezüglich grafischer Repräsentationen. DEViL bietet dagegen spezialisiertere und stärker parametrisierbare grafische Fähigkeiten. So gibt es z.B. Spezialmuster für Bäume VPExplorerTree bzw. VPTree [Sch06b] mit Anbindung an das Graphlayout Werkzeug Dot und diverse Listenmuster. Außerdem sind Funktionen eingebettet um z.B. die Überlappungsfreiheit zu kontrollieren. Die Spezifikationen des Modells mit EMF ist grob mit DEViL vergleichbar. GMF z.B. unterscheidet analog zu DEViL zwischen der semantischen und der editierbaren Struktur, DEViL bietet jedoch weitere Spezialsprachen um die Spezifikation noch weiter zu vereinfachen

Die Entwicklung einer visuellen Sprache [SK03] wurde bereits mit dem VL-Eli [KS02] System, dem Vorgänger von DEViL, in anderen Projekten erfolgreich durchgeführt. So wurde in Zusammenarbeit mit Sagem Orga SIMtelligence Designer/J [SPKF02], eine visuelle Sprache zur Anwendungsentwicklung für SIM-Karten, entwickelt.

7 Zusammenfassung

Das Ziel dieser Arbeit war die Modernisierung und Verbesserung des Entwicklungsprozesses von SPS-Schrittketten in industrieller Umgebung. Dazu wurde eine visuelle domänenspezifische Sprache mit dem Werkzeugsystem DEViL für zwei, miteinander kommunizierende, Entwicklergruppen entwickelt und speziell an deren Aufgaben angepasst.

Für den zu entwickelnden Strukturediteor wurden anhand eines strukturierten Entwicklungsprozesses erst die Situation bei Bosch analysiert, sowie die Aufgaben der Benutzer und das Umfeld der Aufgaben untersucht. Die Benutzungsschnittstelle und die DSL wurden für genau die durchgeführten Aufgaben ausgelegt. Durch die verwendeten Werkzeuge bestand der Hauptteil dieser Arbeit aus der Evaluierung der visuellen Sprache und der Benutzungsschnittstelle. Durch wiederholte kontrollierte Experimente, Interviews und Gruppenbesprechungen mit den Benutzern wurde eine für beide Entwicklergruppen geeignete visuelle DSL sowie eine übersichtliche und aufgabenkonforme Benutzungsschnittstelle entworfen. Der dabei entwickelte Schrittkettenkonfigurator entspricht den Vorstellungen der Benutzer. Durch die Einführung des Schrittkettenkonfigurators wird der Entwicklungsprozess der Schrittketten beschleunigt und vereinfacht.

Durch den in dieser Arbeit entwickelten Schrittkettenkonfigurator wird dem Werkzeugsystem DEViL und dem Vorantreiben dieser Arbeit bei Bosch sehr großes Interesse entgegen gebracht. Obwohl der Schrittkettenkonfigurator alle zu Beginn dieser Arbeit gestellten Anforderungen erfüllt, wurde erst während der Entwicklung die Ausbaufähigkeit des Systems deutlich. So wurden nachträglich zum konformen Ausdrucken der Schrittketten aus dem Schrittkettenkonfigurator zusätzliche Module in DEViL eingebaut und stehen somit auch

allen weiteren mit DEViL generierten Editoren zur Verfügung. Über die systematische Evaluierung des DEViL-Systems und mit ihm erzeugter Sprachimplementierungen wird in [Sch06b] und [SCK07] berichtet. Der Strukturediteor soll in Zukunft weitere Funktionen bekommen und z.B. für Analysen und diverse Simulationszwecke erweitert werden. Einige solcher Erweiterungen sind bereits geplant, z.B. eine visuelle Simulationssicht für Schritte und Aktionen. Das gesamte Konzept hat bei Bosch großen Zuspruch gefunden.

Literatur

[Aue89] A. Auer. *SPS - Aufbau und Programmierung 2., überarb. Aufl.* Huethig, 1989.

[Bar98] Roswitha Bardohl. GenGed: A Generic Graphical Editor for Visual Languages based on Algebraic Graph Grammars. In *1998 IEEE Symp. on Visual Lang.*, Seiten 48–55, September 1998.

[BSM+03] Frank Budinsky, David Steinberg, Ed Merks, Ray Ellersick und Timothy Grose. *Eclipse Modeling Framework*. Addison Wesley, aug 2003.

[Con02] MetaCase Consulting. *MetaEdit+ User's Guide*, 2002. http://www.metacase.com/.

[Eli] *Eli Website*. http://www.uni-paderborn.de/fachbereich/AG/agkastens/eli_home.html.

[Ind] *IndraWorks von Rexroth*. http://www.boschrexroth.com/.

[Inf03] Deutsche Kommission Elektrotechnik Elektronik Informatik. *IEC 61131-3 Speicherprogrammierbare Steuerungen Teil 3*, 2003. Programmiersprachen Deutsche Fassung EN 61131-3:2003.

[KS02] Uwe Kastens und Carsten Schmidt. VL-Eli: A Generator for Visual Languages. In *Proceedings of Second Workshop on Language Descriptions, Tools and Applications (LDTA'02)*, number 2027 in Electronic Notes in Theoretical Computer Science, Grenoble, France, 2002. Band 65, Elsevier Science Publishers.

[Sch06a] Carsten Schmidt. DEViL Homepage, 2006. http://ag-kastens.uni-paderborn.de/forschung/devil/.

[Sch06b] Carsten Schmidt. *Generierung von Strukturediteoren für anspruchsvolle visuelle Sprachen*. Dissertation, 2006.

[SCK07] Carsten Schmidt, Bastian Cramer und Uwe Kastens. Usability Evaluation of a System for Implementation of Visual Languages. *In Symposium on Visual Languages and Human-Centric Computing, IEEE Computer Society Press*, Seiten S. 231– 238, September 2007.

[SK03] Carsten Schmidt und Uwe Kastens. Implementation of visual languages using pattern-based specifications. *Software - Practice and Experience*, 33:1471–1505, Dezember 2003.

[SPKF02] Carsten Schmidt, Peter Pfahler, Uwe Kastens und Carsten Fischer. SIMtelligence Designer/J: A Visual Language to Specify SIM Toolkit Applications. In *Proceedings of Second Workshop on Domain Specific Visual Languages (OOPSLA 2002)*, Seattle, WA, USA 2002, 2002.