Spezifikationen intelligenter machen

Automatische Verarbeitung von smarten Requirements

Bei der heutigen Dynamik von Entwicklungsprojekten ist die Umsetzung von Requirements in den entsprechenden Testfall oft zu langsam oder benötigt zu viele Ressourcen. Für mehr Effizienz sorgt die auf Basis des „Keyword-Driven Testing“ entwickelte formale Sprache Piro, die sich nicht nur zur Erstellung von Testspezifikationen, sondern auch für smarte Systembeschreibungen nutzen lässt.

Autoren: Peter Nicke, Thiemo Frank

Bereits heute stellen Entwicklungsprojekte aufgrund ihrer immer größeren Komplexität hohe Anforderungen an die Requirements- und Testspezifikationen. Eine wesentliche Rolle spielt dabei die Traceability zwischen Kunden-, System- und Softwareanforderung beziehungsweise dem Detailed Design und von da in die jeweiligen Testspezifikationen, sowie zu den dazwischenliegenden Architekturschichten.

In den etablierten Prozessbeschreibungen wie dem V-Modell oder Automotive Spice ist die Traceability als Rückverfolgbarkeit definiert, welche oft durch „einfaches“ Verlinken zwischen den Spezifikationen realisiert wird. Dadurch wird ein struktureller Zusammenhang sichergestellt. Das „Stille-Post“-Problem, also die Frage, ob zwei verlinkte Anforderungen/Testspezifikationen inhaltlich denselben Sachverhalt beschreiben, findet hierbei keine Berücksichtigung. Konsistenz muss durch fehleranfällige Reviews nachgewiesen werden.

Bei der heutigen Dynamik von Entwicklungsprojekten reagieren klassische Entwicklungsprojekte zudem recht träge. Speziell die Umsetzung einer Anforderung in den entsprechenden Testfall ist schon bei minimalen Änderungen oft zu langsam oder benötigt zu viele Ressourcen, um dem dynamischen Entwicklungsumfeld gerecht zu werden. Dies führt zur Diskrepanz zwischen Softwareimplementierung und den zugehörigen Testfällen.

Die Einführung einer wohldefinierten „Requirements Specification Language“ (zum Beispiel TTCN-3), die sowohl in der Implementierungs- als auch in der Testphase Anwendung findet, lässt sich nutzen, um die Spezifikationen auf inhaltliche Konsistenz zu prüfen. Jedoch weist schon eine IEEE-Spezifikation auf Probleme beim Erlernen und Verstehen einer solchen Sprache hin: „One disadvantage in the use of such languages is the length of time required to learn them. Also, many non-technical users find them unintelligible.“

Piro – Phrase in, requirement out

Dagegen kombiniert die formale Sprache „Piro“ (Phrase in, requirement out) etablierte Syntax-Regeln mit einem innovativen Ansatz für die Semantik von Spezifikationen – ohne deren Lesbarkeit zu verschlechtern.

Grundlegend für Piro ist das in der ISO-29119 Part5 beschriebene Verfahren des „Keyword-Driven Testing“ (Bild 1). Piro erweitert dieses Konzept um einen allgemeingültigen Realisierungsvorschlag für die Schlüsselwörter, welche in Piro als Phrasen bezeichnet werden. Ein objektorientierter Ansatz verhilft dabei zu Eindeutigkeit und macht die Begrifflichkeiten verständlicher.

Mithilfe dieser Phrasen lassen sich unter anderem Systemzustände, Signale aber auch Tätigkeiten und Abfolgen von Handlungen leicht und eindeutig beschreiben. Der Ansatz wurde dahingehend optimiert, dass Piro nicht nur zur Erstellung von Testspezifikationen, sondern bereits bei der Anforderungsspezifikation genutzt werden kann. Auch hier lag der Fokus auf der Lesbarkeit beziehungsweise einfachen Anwendbarkeit für den Benutzer. Piro lässt sich ohne größere Schulungsmaßnahmen

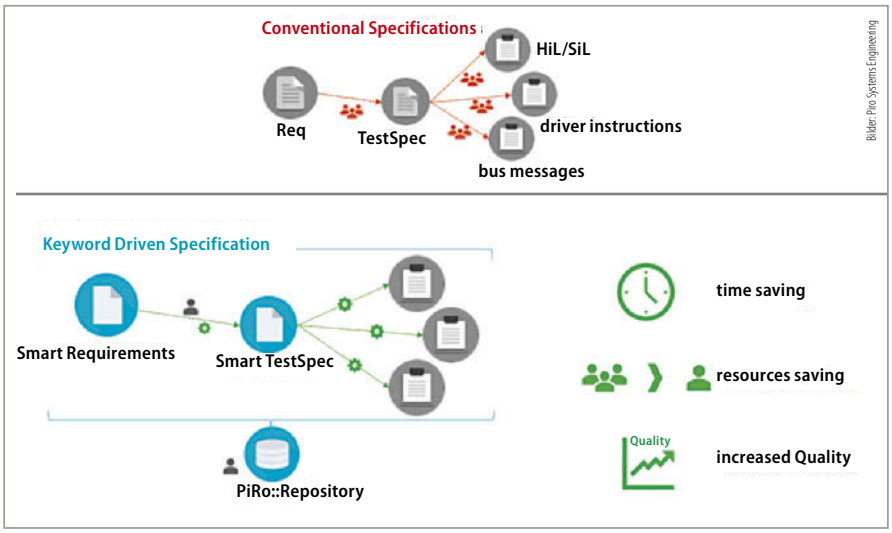

Bild 1: Vergleich einer konventionellen und einer Keyword-Driven Specification.

oder spezielles Expertenwissen direkt und effizient anwenden.

Syntax und Semantik

Piro unterscheidet sich gegenüber anderen formalen Sprachen dadurch, dass der Wortschatz nicht eingeschränkt ist und durch jeden Benutzer selbst erweitert werden kann. Dadurch wird Piro zu einer lebendigen Sprache, die sich ständig selbst weiterentwickelt und verbessert.

Eine Datenbank im Hintergrund lässt sich mit beliebigen Keywords inklusive der entsprechenden Übersetzung in die jeweiligen Domains befüllen. Solche Domains können beispielsweise „Fahreranweisungen", „HiL-/SIL-Prüfskripte", „Signaldefinitionen" oder „Zustände" sein.

Neben dem eigentlichen Keyword besteht eine Piro-Phrase aus den folgenden optionalen Komponenten:

- Klasse (zum Beispiel Zustand, Kodierung, Parameter….)
- System (zum Beispiel Steuergerät xyz, Fahrzeug…)
- Argument (zum Beispiel der Wert einer Variable)

Damit ergibt sich die maximale Ausbaustufe einer Piro-Phrase wie folgt:

Class::System::KeyWord[Argument]

Ein gewünschter Lenkwinkel, der als Kodierwert vorgehalten werden soll, lässt sich beispielsweise wie folgt beschreiben: Coding::EPS::SteerAngleThreshold[20°] .

Der Klasse selber können nun auch Eigenschaften mitgegeben beziehungsweise spezifiziert werden. Das System EPS etwa beschreibt dabei, dass es sich um eine Kodierung innerhalb des EPS-Steuergerätes (Electrical Power Steering) handelt und definiert das Schlüsselwort damit eindeutig.

Das während der Spezifikationsphase entstandene Repositorium lässt sich einfach über Projektteams, Abteilungen und externe Kooperationspartner zusammen mit der eigentlichen Spezifikation verteilen. Piro versteht sich als Metamodell, sodass die konkrete Syntax und Semantik an die individuellen Bedürfnisse des Projektes angepasst werden kann.

Eck-DATEN

Die Komplexität von Fahrzeugsystemen steigt mit jeder neuen Generation exponentiell und die Handhabung der entsprechenden Systembeschreibungen ist ohne intelligente Auswertung nur noch schwer handhabbar. Der mit Piro vertretene Ansatz erfüllt die folgenden Anforderungen an ein Requirements-Management:

- einfach lesbare/verständliche Anforderungen (Qualitätsstandards nach ISO/IEC/IEEE-29148)
- semi- beziehungsweise vollautomatische Erstellung von Testspezifikationen
- automatische Qualitätsbewertung der Anforderungen
- automatische Konsistenzprüfung

Dieser Ansatz wurde bei Daimler im Bereich der Assistenzsysteme etabliert und sorgt inzwischen für ein effizientes Arbeiten im internationalen Team.

Entwicklungsprozess

Mit Piro wird es möglich, ein Requirement in eine abstrakte Piro-Testspezifikation zu überführen und daraus automatisiert Testfälle für die unterschiedlichen Domains zu generieren. Der Transfer vom Requirement zur entsprechenden Testspezifikation wird durch die Wiederverwendbarkeit der jeweiligen Phrasen unterstützt. Es lässt sich eine eineindeutige Beziehung herstellen. Selbst eine vollkommen automatische Übersetzung funktionaler Anforderungen ist mithilfe von einigen wenigen Requirements-Design-Regeln möglich.

Wird innerhalb eines Lastenheftes eine Eingangsvariable wie etwa ein Signal geändert, lassen sich danach „auf Knopfdruck" die entsprechenden Testfälle generieren – ohne Zeitverzug und ohne Ressourcenaufwendungen. Piro dient außerdem als Grundlage, um die Qualität von Requirements und Testspezifikationen bewerten zu können. Auf Basis der Syntax und Semantik werden verschiedene KPIs zur Bewertung eingeführt. Dies hilft dem Anwender bei der Optimierung von Spezifikationen, der Vermeidung von Inkonsistenzen und dem Projektmanagement zur Bewertung des Projektstatus.

Glaubt man der Chaos-Studie der Standish Group, so stellt neben den „unklaren Anforderungen" der in der Regel schwergewichtige Change-Prozesse eine der Hauptursachen für das Scheitern von Softwareprojekten dar. Piro erleichtert den Change-Prozess signifikant, da einfache Änderungen eines Requirements sofort auch in den Testspezifikationen und den entsprechenden Testimplementierungen umgesetzt werden. Piro macht das V-Model hier im wahrsten Sinne schlanker. Selbst bei größeren Änderungen kann in der Regel auf bestehende Definitionen zurückgegriffen werden. Der Ansatz der Keyword Driven Specification ist damit prädestiniert, um ein agiles Umfeld zu unterstützen.

Den Schlüsselwörtern lassen sich Übersetzungen in diverse Spezifikationsdomains zuordnen. So ist es möglich, aus einer allgemeingültigen Testspezifikation automatisiert in HiL-Prüfskripte, Fahreranweisungen oder eine Signalwelt (Flexray, Ethernet…) zu übersetzen. Da eine manuelle Umsetzung in die einzelnen Domains hinfällig ist, können Kosten gespart, Fehler vermieden und die Entwicklungsdyna-

mik erhöht werden. Ähnlich wie bei etablierten Codegeneratoren lassen sich die Struktur der Testscript-Implementierung standardisieren, Programmierregeln festlegen und Kommentare automatisch zufügen. Entsprechend kann die Tracebility zwischen Testspezifikation und der Testimplementierung automatisiert sichergestellt werden (bidirektional).

Tooling

Da der Sprachschatz in komplexen Projekten schnell unübersichtlich wird, ist eine Tool-Unterstützung empfehlenswert. Um die Wiederverwendbarkeit der Phrasen zu gewährleisten, können jeder Phrase weitere Attribute wie etwa Beschreibungen, Suchbegriffe und Ähnliches mitgegeben werden. Durch ein Autovervollständigungstool, das an die Datenbank angeschlossen ist, lässt sich leicht das korrekte Schlüsselwort finden. Ein Wizard hilft dann gegebenenfalls noch bei der Befüllung der Argumente.

Die Akzeptanz einer einheitlichen Spezifikationssprache in komplexen Projekten steht und fällt mit dem Einsatz des richtigen Toolings. Hier ist es elementar, dass die Arbeitsabläufe eines Entwicklers so wenig wie möglich verändert werden und die Anwendung der Sprache keinen zusätzlichen Arbeitsaufwand erzeugt. Für Piro wurde daher eine Möglichkeit gesucht, in der sich ein Tooling nahtlos in die bereits vorhandene Arbeitsumgebung einfügt und sich applikationsunabhängig nutzen lässt. Um dies zu erreichen, steht ein Wizard zur Verfügung, der sich für die Dauer des Spezifizierens über die eigentliche Applikation legt. Der Wizard selbst ist ein Autovervollständigungsprogramm, das bei der Eingabe des Nutzers bereits mögliche Phrasen vorschlägt.

Zusätzlich steht eine Suchfunktion für Phrasen und die Anzeige möglicher Anwendungsbeispiele zur Verfügung. Nach der Spezifikation des Requirements überträgt der Wizard das neue Requirement automatisch an die Zielapplikation und wechselt solange in den Hintergrund, bis der Anwender den Wizard wieder bequem über eine Tastenkombination aufruft. Die Nutzung des Tooling bleibt jedoch optional, sodass jeder Nutzer für sich selber entscheiden kann, wie er damit arbeiten will.

Konkrete Anwendungen

Das beschriebene Verfahren der „Keyword Driven Specification“ stellt eine inhaltliche Konsistenz zwischen Requirement, Testspezifikation und Testimplementierung sicher. Das Verfahren wurde erfolgreich im Bereich kamerabasierte Assistenzsysteme bei Daimler eingesetzt und unterstützt seitdem bei der effizienten und korrekten Umsetzung von Requirements und Testfällen in die entsprechenden Testscripte. Kommunikations- und Implementierungsprobleme des internationalen Entwicklungsteams ließen sich minimieren und unnötige Aufwände reduzieren.

Durch die leicht verständliche Syntax erhöht sich die Qualität der Spezifikationen mit der ersten Phrase und ermöglicht eine einheitliche Testfallbeschreibung innerhalb eines Projektes und darüber hinaus. Des Weiteren ist Piro ideal dazu geeignet, Modelle (etwa UML oder SysML) und deren Artefakte zu beschreiben und das Testing der Modelle zu automatisieren. Da Piro als Spezifikationssprache entwickelt wurde, genügt es den Qualitätsanforderungen der ISO/IEC/IEEE-29148. Dies leistet aktuell keine der vorhandenen alternativen UML-Beschreibungssprachen wie OCL, ALF, FUML oder ähnliche.

Durch die Wiederverwendbarkeit der Phrasen, die Qualitätssteigerung der Spezifikationen und die vorgestellte Möglichkeit, Anforderungen in Testspezifikationen und deren domainspezifische Sprache zu übersetzen, ergibt sich ein hohes Kostensparpotenzial für das Projekt. Die Kommunikation zwischen den einzelnen Stakeholdern wird optimiert und Projektrisiken reduziert.

Darüber hinaus führt die leichte Anwendbarkeit dazu, dass sich Spezifikationen schneller erstellen lassen als bisher. Die sich ständig wiederholende Struktur in Kombination mit den schon definierten Phrasen erlaubt eine effiziente Arbeitsweise. Es muss nicht bei jeder neuen Anforderung über die Art und Weise diese zu spezifizieren nachgedacht werden. Der beschriebene Ansatz eignet sich vor allem für Spezifikationen mit höherem Abstraktionslevels. Spätestens im Detailed Design wird die Syntax/Semantik der späteren Implementierung in den Vordergrund rücken *(ku)* ■

Autoren


Bild: Piro Systems Engineering

Peter Nicke
Geschäftsführer bei
Piro Systems Engineering


Bild: Daimler

Thiemo Frank
Funktionsentwickler bei Daimler