Kapitel 6

Ausblick

Um die hier begonnenen Arbeiten abschließen zu können, sind eine Reihe von weiteren Schritten notwendig, die sich vor allem auf die Vervollständigung des Tests und der Fortsetzung des Reverse Engineering der Motorenkomponente beziehen. Daneben sind auch weitere Arbeiten zur Verbesserung und Verallgemeinerung des Testsystems sinnvoll.

Vervollständigung der Tests (1)
Entwurf von Testpaketen für die noch nicht getesteten Funktionen: “F3 Motorparameter einstellen”, “F6.1 Optimieren eines C-812er Motors”, “F6.2 Optimieren eines C-832er Motors”, “F4 Verfahren nach Encoder-Position” sowie “F1.4 Umrechnung in physikalische Einheiten”.

Zum Test der durch Dialoge realisierten Funktionen kann eventuell die in Entwicklung befindliche Fernsteuerungs-Software von J. Hanisch und J. Letztel verwendet werden.

Bestimmung der *white box*-Testüberdeckung
Zur Beurteilung der Vollständigkeit des Tests ist eine genaue Angabe zur Überdeckung des Quelltextes durch die entworfenen Testfälle erforderlich. Dies im Besonderen bei Reverse Engineering Projekten, bei denen von unvollständiger Kenntnis der durch den Quelltext realisierten Funktionen ausgegangen werden muss.

Vervollständigung der Tests (2)
Entwurf weiterer Testfälle zur Erreichung eines bestimmten *white box*-Überdeckungsgrades. Die mit *white box*-Verfahren gefundenen Testfälle sollten nach Möglichkeit als *black box*-Testfälle in die bereits vorhandenen Testpakete integriert werden. Eventuell kann an dieser Stelle begonnen werden nicht erreichbaren Code zu entfernen.

Korrektur der Fehler
Um sicherzustellen, dass Korrekturen und sonstige Veränderungen im Quelltext das Verhalten der Motorenkomponente nicht unerwünscht ändern, sollten theoretisch Veränderung erst durchgeführt werden, wenn Testfälle für den Regressionstest in ausreichender Vollständigkeit vorhanden sind.

Da bis zu diesem Stand aber wahrscheinlich zu viel Zeit vergehen wird, muss die Fehlerbeseitigung wohl parallel zur Weiterentwicklung des Tests erfolgen. Dies vor allem bei schwer wiegenden Fehlern und bei Fehlern, deren

Ursache gut zu lokalisieren und ausreichend verstanden ist. Später entwickelte Testfälle können dann (theoretisch) immer noch auf die verschiedenen Versionen der Komponente angewendet werden.

Vervollständigung der Verhaltensspezifikation
Parallel zu den anderen Arbeiten muss die Verhaltensspezifikation vervollständigt werden, in der immer noch Abschnitte offen bzw. unvollständig sind.

Hier sind Überlegungen zu der Frage sinnvoll, wie die während der Testentwicklung entstandenen Dokumente in die Beschreibung des Ist- bzw. des Soll-Verhaltens einfließen können. Hier scheinen vor allem die Klassifikationsbäume und die Soll-Daten-Dokumente von Interesse, die im Prinzip Beschreibungen des Verhaltens der zu testenden Komponente darstellen.

Dokumentation der Implementation der Motorenkomponente
Ursprünglich sollte schon im Rahmen dieser Arbeit ein Teil der Dokumentation entstehen, ist aber wegen des Umfanges nicht mehr aufgenommen worden. Es steht bisher nur eine Modell-Datei für das Softwareentwicklungswerkzeug Rational Rose zur Verfügung, in der Teile der Komponente dokumentiert wurden.

Verallgemeinerung des Testsystems
Das Testsystem könnte langfristig derart verallgemeinert werden, dass die einzelnen Schritte der Testdurchführung: Testfallentwicklung und Testskripterzeugung sowie Testausführung und -auswertung stärker voneinander abgegrenzt werden, und so wahlweise durch unterschiedliche Werkzeuge durchgeführt werden können ohne dass dadurch das Testsystem insgesamt verlassen wird.

Verbesserungen des Vergleichers
Verallgemeinerte Implementation des Vergleichers zur Unterstützung spezieller Datentypen der Ausgabedaten, Unterstützung verschiedener Unschärfe- bzw. Gleichheits-Definitionen, und zur besseren Analyse der Unterschiede zwischen Soll- und Ist-Daten.

Z.B. erfolgt die Analyse der Soll- und Ist-Daten-Unterschiede in der vorliegenden Implementation mit Hilfe des Werkzeuges `diff`, was nur bei einfachen Strukturen der Daten gut funktioniert.

Integration Plattform-abhängiger Soll-Daten
Mitunter ist das Verhalten eines zu testenden Systems Plattform- oder Betriebssystem-abhängig. Im Fall der Motorenkomponente werden z.B. von der Implementation für die `C-812ISA`er-Motoren Betriebssystem-abhängige Basisadressen für den Direktspeicher verwendet. Die zur Kommunikation verwendeten Adressen sind Teil der Ausgaben und somit sind diese Betriebssystemabhängig. Im Moment erfolgt eine Anpassung der Soll-Daten an das Betriebssystem auf dem der Test durchgeführt wird durch einen einfachen Suchen-und-Ersetzen-Mechanismus.

Langfristig ist sicher eine Integration solcher Ausgabe-Varianten in die Soll-Daten wünschenswert. Die Lösung könnte analog zu der der Integration von *baseline*- und *targetline*-Daten erfolgen.

`make` ersetzen
Das regelbasierte Softwareentwicklungswerkzeug `make` hat sich im Wesentlichen bewährt. Ein Problem aber ist, dass die Entscheidungen darüber, wel-

che Teile des Tests erneut auszuführen sind, von diesem Werkzeug aufgrund der Datumsstempel der Dateien gefällt werden. Dies ist dann problematisch, wenn mehrere Dateien aus einer generiert werden, wie es in diesem Testsystem bei den Testskripten de Fall ist, die innerhalb eines Testpakets alle aus einer CTE-Datei erzeugt werden. Wird nun ein weiterer Testfall in ein Paket aufgenommen, oder wird auch nur Dokumentation in der CTE-Datei bearbeitet, so sind gemäß der `make`-Logik alle Testfälle des Paketes erneut durchzuführen.

Hier wäre also die Suche nach einen `make`-ähnlichen Werkzeug notwendig, das für die problematische Entscheidung andere Kriterien als die Datumsstempel verwendet. Eine mögliche Alternative könnte das Werkzeug `cons` – *A Software Construction System* – sein, das für die Bestimmung von Veränderungen in Dateien MD5-Prüfsummen verwendet.

Dokumentation verbessern

Die Anzahl der Testfälle in einem Testpaket kann schnell sehr groß werden. Bei Klassifikationsbäumen mit mehreren Verfeinerungen und in Testpaketen, die Testsequenzen enthalten, ist es relativ schwierig über größere Mengen von Testfällen bzw. -schritten den Überblick zu behalten, was die Wartung der Testpakete erschwert. Hier ist die Realisierung einer übersichtlichen Dokumentation einer großen Anzahl von Testfällen notwendig Ebenso würden verschiedene Analysen der Kombinationstabelle die Pflege der Testpakete verbessern.