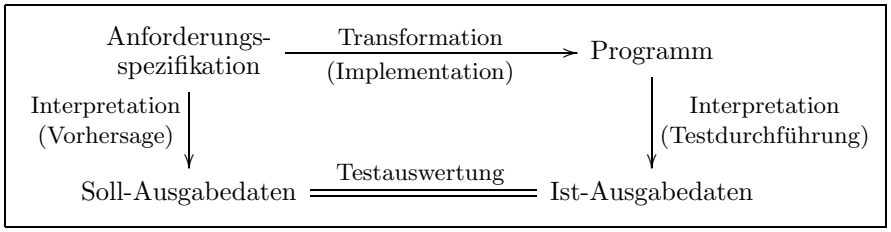
Kapitel 2

Softwaretest in einer *Reverse Engineering*-Situation

Bei der Planung und der Vorbereitung des Tests, stellte sich sehr schnell die Frage: Was bedeutet Test in einem *Reverse Engineering (RE)*-Projekt?

Test in einem *Forward Engineering* Projekt Im normalen Fall eines *Forward Engineering (FE)*-Projektes bedeutet Test, zu überprüfen, ob die zu Beginn des Projektes oder einer Projektphase in einer Spezifikation zusammengetragenen Anforderungen durch das entstandene Programm erfüllt werden. Ausgangspunkt für die Testentwicklung ist demzufolge die funktionale Spezifikation. Aus ihr lassen sich *black box*-Testfälle entwerfen und nach der Testdatenauswahl die Soll-Ergebnisse des Tests vorhersagen (s. Abb. 2.1). Die Vollständigkeit des Tests wird durch die Anwendung aller spezifizierten Funktionen in den Testfällen erzielt und kann durch die Verwendung von *white box*-Verfahren überprüft und ergänzt werden. Der Test kann parallel zum Programm entwickelt werden. Wenn das Programm fertig ist, kann getestet werden, ob es der Spezifikation entspricht.

Mit anderen Worten, der Test überprüft die Korrektheit der Transformation der (Anforderungs)Spezifikation in das Progamm, in dem die Ergebnisse, die sich bei der Interpretation der Spezifikation vorhersagen lassen, mit den Ergebnissen der Programmausführung verglichen werden.



Aus der Äquivalenz der Soll- und Ist-Ergebnisse wird die Korrektheit der Implementation (bzgl. der Testfälle) geschlossen. Unter der Annahme, dass die Spezifikation korrekt ist, d.h. die Anforderungen richtig wiedergibt, kann daher angenommen werden, dass das Programm den Anforderungen entspricht. Die Vollständigkeit der Transformation wird aus der Vollständigkeit der Testfälle

abgeleitet.

Schwierigkeiten in diesem Vorgehen bereitet im wesentlichen die Vorhersage der Soll-Ergebnisse. Während die Testdurchführung und zu großen Teilen auch die Testevaluierung automatisierbar sind, ist die Festlegung der zu erwartenden Ergebnisse i.d.R. ein manueller Vorgang. Dies liegt daran, dass in den meisten Fällen die Spezifikationen informal sind und sich damit einer automatisierten Interpretation weitgehend verschließen. Auch ist es fraglich, ob diese Spezifikationen die Detailtiefe erreichen, die notwendig ist, das Verhalten vollständig vorherzusagen.

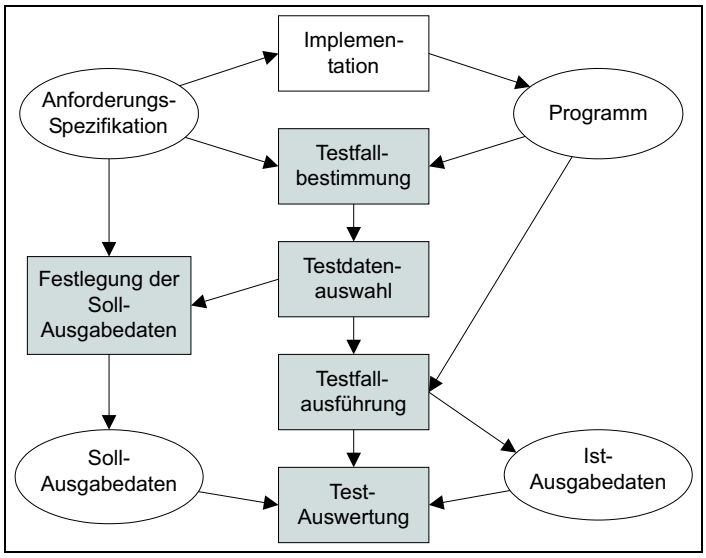
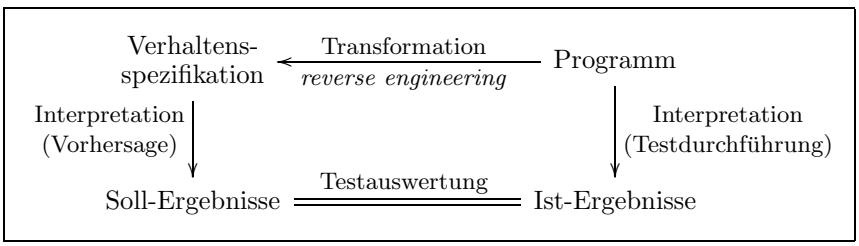


Abbildung 2.1: Testaktivitäten beim *Forward Engineering* (vgl. [Weg00])

Test in einem *Reverse Engineering* Projekt Während also beim FE die Spezifikation den Ausgangspunkt der Implementation und des Tests darstellt, steht beim RE am Anfang das Programm. Aus diesem kann, durch Beobachtung des Verhaltens und Untersuchung des Quelltextes, eine (Verhaltens)Spezifikation abgeleitet werden. Aus der Spezifikation kann analog zum FE-Prozess eine Vorhersage bezüglich der zu erwartenden Ergebnisse gemacht werden. Desweiteren können, wie beim FE-Prozess, durch die Ausführung des Programmes Ist-Ergebnisse gewonnen werden.



Aus der Übereinstimmung der vorhergesagten Soll- und der Ist-Ausgabedaten kann darauf geschlossen werden, dass die Spezifikation das Verhalten des Pro-

grammes korrekt wiedergibt. Im Falle eines korrekten Programmes ist ein Ziel des RE damit erreicht. Es ist eine Spezifikation erstellt worden, es ist sichergestellt, dass das Programm dieser entspricht und man hat Testfälle, die den Regressionstest ermöglichen.

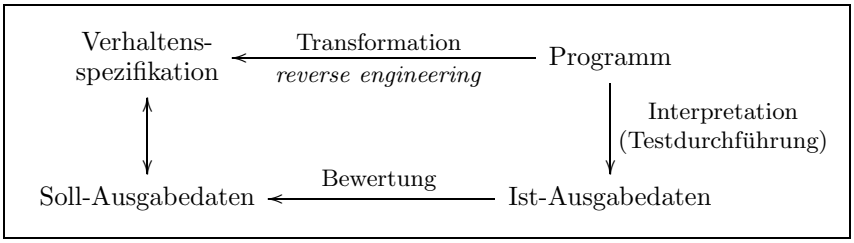
Dieses Vorgehen wirft im Bezug auf unsere Aufgabenstellung zwei Probleme auf:

Zum einen (und vor allem) das bereits angesprochene Problem der informalen Spezifikationen und der Ergebnisvorhersage. Die während der ersten RE-Schritte entstandenen Versionen der (Verhaltens)Spezifikation für die zu testende Motorenkomponente haben nicht die Detailgenauigkeit erreicht, die notwendig wäre, das Verhalten der Komponente v.a. in Bezug auf die Ansteuerung der Motoren vorherzusagen. Aber es schien notwendig, gerade diesen Aspekt des Verhaltens besonders zu berücksichtigen. Auch besteht ein gewisses Problem darin, dass derjenige, der die Spezifikation aus dem Programm abgeleitet hat, auch derjenige sein soll, der zur Überprüfung der Spezifikation das Programmverhalten aus dieser vorhersagt.

Der andere Punkt ist das Problem, dass das zu testende Programm bekanntermaßen fehlerhaft ist. D.h., dass das Verfahren eine Spezifikation liefert, die das korrekte Abbild eines unkorrekten Programmes ist. Das kann im Sinne einer Bestandsaufnahme durchaus Sinn machen, hilft aber nicht das fehlerhafte Verhalten vom korrekten zu trennen.

Im Rahmen des „Softwaresanierungs“-Projektes ist die Problematik „Was sollte die im RE zu erstellende Spezifikation enthalten?“ öfter diskutiert worden, und hat zu der Unterscheidung zwischen *Anforderungsspezifikation* und *Verhaltensspezifikation* geführt; das erste als Ausgangspunkt des FE und das zweite als Ergebnis des RE. Während eine Verhaltensspezifikation den Ist-Zustand eines Projektes, einschließlich der fehlerhaften und unvollständigen Funktionen und der Änderungswünsche, wiedergibt, beschreibt die Anforderungsspezifikation ausschließlich einen Ziel-Zustand. Die Unterscheidung der Dokumente soll die klare Trennung von Wunsch und Realität ermöglichen und sicherstellen. Es bleibt aber die Frage offen, wie weit man in der Beschreibung des Ist-Zustandes geht — ob eine Verhaltensspezifikation eine exakte Beschreibung eines bereits als falsch erkannten Verhaltens enthalten sollte, oder ob sie nur den Fehler benennen und das Soll-Verhalten beschreiben sollte.

Das gewählte Vorgehensmodell Wegen der genannten Probleme ist nicht der beschriebene Weg für das RE gewählt worden, sondern ein Weg, der die Testentwicklung stärker in den Prozess der Wissenswiedergewinnung und der Bewertung des Programmes einbettet; der das Ist-Verhalten konserviert, das bewertete Ist-Verhalten zur Grundlage der Festlegung des Soll-Verhaltens macht und dieses mit der Spezifikation in Übereinstimmung bringt.



Im folgenden soll das konkrete Vorgehen schrittweise beschrieben werden. Abb. 2.2 stellt dieses Vorgehen schematisch dar.

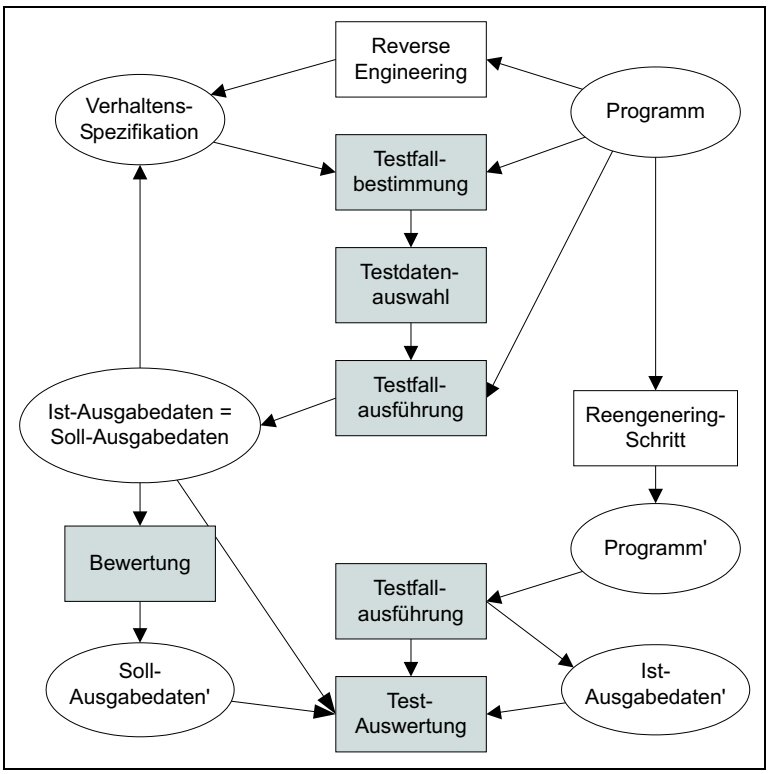


Abbildung 2.2: Testaktivitäten beim *Reverse Engineering*

Verhaltensspezifikation
Am Anfang steht die Sammlung und Beschreibung der Funktionen in der Verhaltensspezifikation. Das Programm wird ausgeführt und das Verhalten wird beobachtet und beschrieben. Neben dem beobachteten Verhalten werden in der Spezifikation auch Erwartungen formuliert, die aus dem Wissen über den Anwendungsbereich resultieren.

Ausgangspunkt für das Finden von Funktionen sind zum einen Anwendungsfälle, die sich aus dem Nutzerinterface ergeben, zum anderen sind aber auch Quelltextanalysen notwendig. So kann z.B. die Funktion „Referenzpunktlauf“ der Motorenkomponente durch die Auswertung des entsprechenden Dialoges gefunden und beschrieben werden. Dagegen aber ist die Funktion „Behandlung des Motorspiels“ für den Anwender transparent und kann nur in den Quelltexten gefunden und verstanden werden.

Der erste Arbeitsschritt ist also kein spezieller Test-Schritt, sondern ein normaler Arbeitsschritt des RE.

Testentwicklung
Ausgehend von der Verhaltensspezifikation können für einzelne Funktionen systematisch *black box*-Testfälle entworfen und entsprechende Testdaten aus-

gewählt werden. Dabei wird angestrebt alle Aspekte der zu testenden Funktionen zu erfassen und in den Testfällen vollständig zu verwenden.

Die Testausführung liefert eine Beschreibung des Verhaltens. Diese Beschreibung konserviert das Ist-Verhalten und kann für den Regressionstest verwendet werden. Die Ausgabedaten, die das Ist-Verhalten darstellen, sind damit Soll-Ausgabedaten im Sinne einer Verhaltensspezifikation des existierenden Systems.

Bewertung
Das Ist-Verhalten, das potentiell Fehler enthält, wird im nächsten Schritt einer *Bewertung* unterzogen. Dabei wird das tatsächliche Verhalten mit dem in der Verhaltensspezifikation beschriebenen verglichen. Werden Unterschiede gefunden ist entweder die Spezifikation der Funktion falsch bzw. die Funktion ist noch nicht richtig verstanden oder aber es wurde ein echter Fehler im existierenden Programm gefunden.

Andererseits kann auch die Übereinstimmung des Ist-Verhaltens mit der Verhaltensspezifikation eine Fehlersituation darstellen, falls eine fehlerhafte Funktion des existierenden Programmes in die Verhaltensspezifikation übernommen wurde.

Die Entscheidung darüber, ob ein Fehler vorliegt oder nicht, ist unter Umständen problematisch, da stets eine gewisse Unsicherheit darüber angebracht ist, ob der Fehler wirklich im Programm oder aber im Verständnis des Programmes zu suchen ist. Die Entscheidung wird i.d.R. wie folgt getroffen:

- Das Verhalten ist *offensichtlich* fehlerhaft; das Programm „stürzt ab“, „hängt sich auf“ oder produziert unmittelbar erkennbare Widersprüche.
- Die *Anwender* identifizieren das Verhalten als falsch entsprechend ihrem Wissen über den Anwendungsbereich.
- Die *Umgebung* des Programmes verhält sich entsprechend; z.B. kann die Hardware oder deren Simulation Signale produzieren, die auf einen Fehler des Programmes schließen lassen.
- Der *Entwickler bzw. Tester* mit seinem speziellen Wissen über Programm und Anwendungsbereich erklärt das Verhalten für falsch.

Liegt kein Fehler des Programmes vor, muss die Spezifikation korrigiert oder vervollständigt werden.

Wenn tatsächlich ein Fehler im Programm gefunden wurde, ist dies in der Spezifikation zu beschreiben. Das korrekte Verhalten bzw. die korrekten Ergebnisse müssen bestimmt werden und die Soll-Daten müssen entsprechend angepasst werden. D.h. die Ist-Ausgabedaten werden überführt in Soll-Ausgabedaten im Sinne einer Anforderungsspezifikation.

Nach diesem Schritt stehen also zwei Typen von Soll-Daten für den Test zur Verfügung: die Soll-Daten, die den aktuellen oder Ausgangs-Zustand konservieren und die, welche den Ziel-Zustand beschreiben. Das Nebeneinander der beiden Soll-Daten-Typen dient auch als Absicherung gegen Irrtümer in der Beurteilung von Fehlern.

Überprüfung der Vollständigkeit
Im nächsten Schritt geht es darum, sicherzustellen, dass die während des RE erstellte Spezifikation alle Funktionen des untersuchten Programmes enthält. Dies wird indirekt durch die Vollständigkeit der Testfälle bestimmt. Dazu wird nach Durchführung aller aus der Spezifikation abgeleiteten *black box*-Testfälle, mit Hilfe von *white box*-Verfahren überprüft, welche Teile des Quelltextes verwendet worden sind und welche nicht[1].

Die Testfälle werden solange vervollständigt, bis ein ausreichendes Maß an Quelltextüberdeckung sichergestellt ist. Was dabei *ausreichend* bedeutet, ist eine Frage des vertretbaren Aufwandes und der zur Verfügung stehenden Werkzeuge. Anweisungsüberdeckung sollte aber zur Sicherstellung der Vollständigkeit wenigstens angestrebt werden.

Nicht überdeckter Quelltext realisiert potentiell noch nicht erkannte Funktionen oder noch nicht erkannte Aspekte bereits erkannter Funktionen. Daher sollte versucht werden, die durch *white box*-Verfahren gefundenen Testfälle als *black box*-Testfälle zu formulieren. Diese Formulierungen führen dann eher zu den notwendigen Ergänzungen in der Spezifikation.

Regressionstest
Der folgende Testschritt – der Regressionstest – ist dann kein spezieller RE-Schritt mehr. Das zu testende Programm ist verändert worden, z.B. wurden Fehler beseitigt oder Umstrukturierungsmaßnahmen vorgenommen. Alle Testfälle werden mit dem neuen Programm durchgeführt. Die Testauswertung erfolgt bezüglich der beiden Soll-Daten-Sätze und testet zum einen, wo sich das Verhalten des Programmes durch die Änderungen überall verändert hat und zum anderen, ob diese Änderungen nun dem Ziel-Verhalten entsprechen oder nicht. Die Soll-Daten der Verhaltensspezifikation sind entsprechend den durchgeführten Änderungen zu aktualisieren.

Es ist wichtig die beschriebene Vorgehensweise als einen iterativen Vorgang zu verstehen, der *schrittweise* die Spezifikation und den Test vervollständigt, da in den Informationsflüssen zwischen den Arbeitsschritten ein Zyklus vorliegt:

- das Verhalten wird während der exemplarischen Ausführung beobachtet und aus dem zu beobachtenden Verhalten wird eine Spezifikation abgeleitet,
- ausgehend von der Spezifikation werden systematisch Testfälle entwickelt,
- aus den Ist-Ergebnissen der Testausführung wird auf das Verhalten zurück geschlossen, die Spezifikation wird vervollständigt und korrigiert,
- und aus den Informationen über die während der Testausführung genutzten Quelltextteile werden neue Testfälle und damit neues Verhalten gefunden.

Die Testentwicklung wird hier als direkter Teil des RE angesehen. Sie dient der zielgerichteten und systematischen Auseinandersetzung mit der aufzubereitenden Software. Sie führt direkt zu Wissen, das sich in der Spezifikation verarbeiten lässt und gleichzeitig zu einem Werkzeug, mit dem die verändernden RE-Schritte kontrolliert und überprüft werden können.

[1] Zu den *white box*-Verfahren (auch Struktur- oder *glass box*-Tests genannt) s. [Bal98, S. 400ff.]