Entwicklung eines Entscheidungsunterstützungssystems für die Planung von Schnittstellentests im Rahmen der Qualitätssicherung von komponentenbasierter Unternehmenssoftware

Franz Rothlauf

Working Paper 6 / 2004

Juli 2004

Working Papers in Information Systems 1

University of Mannheim
Department of Information Systems 1
D-68131 Mannheim/Germany

Phone +49 621 1811691, Fax +49 621 1811692

E-Mail: wifo1@uni-mannheim.de
Internet: http://www.bwl.uni-mannheim.de/wifo1

Entwicklung eines Entscheidungsunterstützungssystems für die Planung von Schnittstellentests im Rahmen der Qualitätssicherung von komponentenbasierter Unternehmenssoftware

Franz Rothlauf

Lehrstuhl für ABWL und Wirtschaftsinformatik I, Universität Mannheim

rothlauf@uni-mannheim.de

Zusammenfassung: Bei der Erstellung von komponentenbasierter Unternehmenssoftware ist es eine Aufgabe der Qualitätssicherung sicherzustellen, dass die Schnittstellen zwischen den einzelnen Komponenten keine Fehler aufweisen. Der vorliegende Beitrag beschreibt die Konzeption und Umsetzung eines Entscheidungsunterstützungssystems (EUS), welches die Qualitätssicherungsabteilung eines Anbieters von Unternehmenssoftware bei der Aufstellung eines Zeitplans für die zentrale Durchführung von Schnittstellentests unterstützt. Wichtigstes Element des EUS ist ein metaheuristikbasiertes Planungssystem, welches dem Entscheider qualitativ gute Zeitpläne vorschlägt. Das Planungssystem berücksichtigt sowohl unterschiedliche Planungsziele, als auch relevante Problemrestriktionen durch eine geeignete Wahl der Problemkodierung, einer geschickten Erzeugung des Zeitplans aus der Problemkodierung, sowie einer geeigneten Initialisierung der Startlösungen.

Schlüsselworte: Reihenfolgeplanung, Schnittstellenplanung, EUS, Qualitätssicherung, komponentenbasierte Softwareentwicklung

1. Einleitung

Bei der Erstellung von komponentenbasierter Unternehmenssoftware stellt sich das Problem, dass im Rahmen der Qualitätssicherung sowohl die Funktionalität der einzelnen Softwarekomponenten, als auch deren Zusammenspiel überprüft werden müssen. Die Funktions- und Qualitätskontrolle der Komponenten wird hierbei in der Regel durch die einzelnen Fachabteilungen selbst durchgeführt und nur die Endkontrolle durch eine zentrale Qualitätssicherungsabteilung vorgenommen. Bei der Überprüfung des korrekten Zusammenspiels der einzelnen Komponenten sind eine Vielzahl von unterschiedlichen Fachabteilungen involviert, welche durch eine dedizierte Qualitätssicherungsabteilung organisiert und koordiniert werden. Hierbei stellt sich die Aufgabe, dass Schnittstellentests an zentraler Stelle durchgeführt werden und ein entsprechender Zeitplan für die Durchführung der Schnittstellentests erstellt werden muss. Das Aufstellen eines derartigen Zeitplans ist aufwendig, da eine Menge unterschiedlicher Ziele existieren und zusätzliche Restriktionen bezüglich der zeitlichen Verfügbarkeit von Ressourcen und Personen sowie Unverträglichkeiten zwischen einzelnen Softwarekomponenten berücksichtigt werden müssen.

Das Problem der Erzeugung von Zeitplänen für die Durchführung von Schnittstellentests ist vergleichbar mit Maschinenbelegungs- und Stundenplanbelegungsproblemen. Derartige Probleme gehören zu den traditionellen Forschungsgebieten der Betriebswirtschaftslehre und insbesondere im Operations Research wurde eine Vielzahl von Modellen und Methoden entwickelt, welche theoretisch optimale Lösungen ermitteln können [Brü95]. Aufgrund der NP-Vollständigkeit der zugrundeliegenden Planungsprobleme sind diese allerdings auf viele komplexe Problemstellungen nicht anwendbar, da der Lösungsaufwand exponentiell mit der Größe des Planungsproblems zunimmt. Aus diesem Grund werden insbesondere in der Wirtschaftsinformatik zunehmend Metaheuristiken zur Lösung komplexer Planungsprobleme eingesetzt [Zäpf94, Sche91, DaWa97, KäTe04]. Mit deren Hilfe können Planungsprobleme gelöst werden, welche klassischen, exakten Methoden nicht zugänglich sind [SWM95, ScMe00]. In Zusammenhang mit den in den letzten Jahrzehnten stark angestiegenen Rechnerleistungen werden metaheuristikbasierte Verfahren mittlerweile auch standardmäßig in elektronischen Leitständen für die Fertigungssteuerung [Kurb99, DaWa97] oder Advanced Planning Systemen wie z. B. von SAP (APO-Suite) oder i2 (logistics optimization) eingesetzt.

Der vorliegende Beitrag stellt die Konzeption und Umsetzung eines EUS für die Planung von Schnittstellentests für komponentenbasierte Unternehmenssoftware vor. Das EUS wurde in Zusammenarbeit mit einem großen deutschen Softwarehersteller im Bereich Steuer- und Buchführungssoftware konzipiert und implementiert und mittlerweile mit Erfolg in der Qualitätssicherung eingesetzt. Das im EUS integrierte Planungssystem erstellt mit Hilfe einer Metaheuristik (Genetische Algorithmen) Zeitpläne für die Durchführung von Schnittstellentests und berücksichtig hierbei unterschiedliche Ziele und Nebenbedingungen. Die verschiedenen Ziele und einzuhaltenden Restriktionen werden durch die Konzeption einer geeigneten Methode zur Kodierung und Bewertung von Zeitplänen sowie durch eine geschickte Erzeugung von Startlösungen für die Metaheuristik berücksichtigt. Im Rahmen des Beitrags wird das entsprechende Planungsproblem beschrieben, mögliche Ansätze zur Lösung des Planungsproblems diskutiert sowie das Fachkonzept, die Umsetzung und die Implementierung des EUS dargestellt.

Im folgenden Abschnitt wird eine genaue Beschreibung des Planungsproblems einschließlich der beim Praxispartner aufgetretenen Nebenbedingungen gegeben. Abschnitt 3 diskutiert mögliche Lösungsansätze für das Problem. Im Hauptteil des Beitrags (Abschnitt 4) wird die Konzeption und Funktionsweise des Planungssystems beschrieben. Es wird dargestellt, wie Testpläne erstellt und bewertet werden und wie das Planungssystem in das EUS integriert ist. Nach einer kurzen Diskussion der Erfahrungen bei der Entwicklung und beim Einsatz des Systems in Abschnitt 5 endet der Beitrag mit einer kurzen Zusammenfassung.

2. Erstellung von Zeitplänen für Schnittstellentests bei komponentenbasierter Unternehmenssoftware

Bei der komponentenbasierten Softwareerstellung wird die Funktionalität der zu erstellenden Software in einzelne Komponenten zerlegt, welche unabhängig voneinander weiterentwickelt werden können. Wichtig bei der Entwicklung von komponentenbasierter Unternehmenssoftware ist, dass verbindliche und klar definierte Schnittstellen zwischen den einzelnen Softwarekomponenten festgelegt werden. Es ist Aufgabe der Qualitätssicherung die Einhaltung der Schnittstellenspezifikati-

onen zu überprüfen und Fehler im Zusammenspiel zwischen unterschiedlichen Softwarekomponenten aufzudecken.

Bei der hier vorliegenden Problemstellung wird der Test der einzelnen Schnittstellen durch die Qualitätssicherung organisiert und entsprechende Spezialisten und Entwickler aus den einzelnen Fachabteilungen hinzugezogen. Die eigentlichen Tests werden zentral bei der Abteilung für Qualitätssicherung zeitlich parallel auf einer vorgegebenen Menge von Testsystemen (ca. 5-8), welche jeweils aus zwei Rechnern bestehen, durchgeführt. Der Test einer Schnittstelle wird immer durch einen oder zwei Ansprechpartner (Tester) aus den jeweiligen Fachabteilungen, welche für die beteiligten Komponenten zuständig sind, durchgeführt. Für die Durchführung der Tests ist ein Zeitplan aufzustellen, welcher festlegt, wann welche Schnittstellentests auf welchen Testsystemen durchgeführt werden sollen. Durch den Zeitplan wird auch bestimmt, welche Ansprechpartner (ca. 20-50 verschiedene) wann welche Schnittstellentests durchführen. Obwohl die einzelnen Schnittstellentests in recht kurzer Zeit durchgeführt werden können (30 bzw. 60 Minuten) ist aufgrund der Vielzahl der zu testenden Schnittellen (üblicherweise im Bereich zwischen 300 und 500) der gesamte Zeitaufwand (ca. 4-7 Tage) für die Durchführung aller Tests hoch. Üblicherweise werden die Schnittstellentests erst kurz vor dem nächsten Softwareveröffentlichungstermin durchgeführt, sodass der vorgegebene Zeitrahmen für die Durchführung der Tests beschränkt ist.

Die folgenden Abschnitte stellen die Anforderungen an ein Planungssystem zum Lösen des Planungsproblems vor. Es wird zuerst auf die unterschiedlichen und teilweise auch konkurrierenden Ziele des Planungsprozesses und anschließend auf Nebenbedingungen bei der Erstellung eines Zeitplans eingegangen.

2.1. Ziele des Planungsprozesses

Im Folgenden werden die konfligierenden Zielsetzungen des Planungsprozesses vorgestellt, welche im Rahmen des Projektes in Zusammenarbeit mit der Qualitätssicherungsabteilung ausgearbeitet wurden.

2.1.1. Zusammenhängende Testzeiträume für einzelne Tester

Bei der Durchführung von Schnittstellentests muss für jede Schnittstelle je ein Ansprechpartner (Tester) für jede der beteiligten Komponenten vor Ort sein. Ziel der Planung soll es sein, dass jeder Tester jeweils eine möglichst große Menge von Tests am Stück, das heißt zeitlich nacheinander, durchführen kann. Die Tests können hierbei auch an unterschiedlichen Testsystemen (alle Testsysteme befinden sich in einem Raum) durchgeführt werden, aber es sollten keine Lücken im Testplan des jeweiligen Ansprechpartners entstehen, da ansonsten immer zusätzliche Wege für den Schnittstellentester anfallen.

2.1.2. Frühzeitiger Test von Komponenten mit Basisfunktionalität

Die zu testenden Komponenten lassen sich in zwei Gruppen unterteilen: Es existieren Basiskomponenten, welche grundlegende Basisfunktionalitäten des Softwaresystems realisieren und von anderen Komponenten wiederverwendet werden. Auf diese Basiskomponenten (z. B. Stammdatenverwaltung) setzen dann Zusatzkomponenten (z. B. Auskunftssystem) auf, welche durch Basiskomponenten zur Verfügung gestellte Funktionalitäten nutzen und zusätzliche Funktionalität zur Verfügung stellen.

Bei der Durchführung von Schnittstellentests sollen Basiskomponenten vor Zusatzkomponenten getestet werden. Falls Basiskomponenten vor Zusatzkomponenten getestet werden, kann eindeutig bestimmt werden, welche Fehler auf Basiskomponenten und welche auf Schnittstellen von Zusatzkomponenten zurückzuführen sind. Falls Basiskomponenten erst nach Zusatzkomponenten getestet werden, können auftretende Fehler nicht mehr eindeutig zugeordnet werden und es entsteht ein höherer Aufwand bei Fehlersuche und Fehlerzuordnung.

2.1.3. Minimierung der Installationszeiten

Für die Durchführung von Schnittstellentests ist es notwendig, dass die jeweils zu testenden Softwarekomponenten auf den Testsystemen installiert werden. Üblicherweise werden die Installationen durch die Abteilung Qualitätssicherung vor der Testphase durchgeführt und für die eigentlichen Tests werden von den Testern die schon installierten Komponentenversionen verwendet. Bei Bedarf müssen allerdings bei der Aufstellung eines Zeitplans zusätzliche Installationszeiten für Komponenten berücksichtigt werden. Die Gründe hierfür liegen z. B. in knappen Entwicklungszeiten von Produkten, in zu erwartenden kurzfristigen Änderungen der Produktversionen oder in einem sehr frühen Beginn der Testphase zu einem Zeitpunkt, in dem noch keine „reifen" Produktversionen existieren. Beim erstmaligen Test einer Schnittstelle auf einem der vorhandenen Testsysteme muss dann die entsprechende Komponente zunächst installiert werden. Die dafür notwendige Zeit muss bei der Erstellung des Zeitplans berücksichtigt werden. Ziel des Planungsprozesses ist es, die Gesamtinstallationszeiten zu minimieren, was z. B. dadurch realisiert werden kann, dass die entsprechenden Komponenten auf einer möglichst geringen Anzahl an Testsystemen installiert und getestet werden.

2.2. Bei der Planung einzuhaltende Nebenbedingungen

Neben der Vorgabe von Planungszielen existieren auch Nebenbedingungen, welche durch ein automatisiertes Planungssystem berücksichtigt werden müssen.

2.2.1. Einhaltung des vorgegebenen Gesamtzeitraums für die Durchführung der Schnittstellentests

Eine wichtige Nebenbedingung bei der Aufstellung eines Zeitplans ist die Einhaltung des vorgegebenen Gesamtzeitraums (in Tagen gemessen), welcher für die Durchführung aller Schnittstellentests angesetzt wird. Üblicherweise werden nur wenige Tage für die Durchführung der Tests eingeplant, da die Tests möglichst schnell abgeschlossen werden sollen, um genügend Zeit für die Behebung der bei den Tests aufgedeckten Probleme zu haben.

2.2.2. Keine gleichzeitige Durchführung von verschiedenen Schnittstellentests auf einem Testsystem

Auf jedem der unterschiedlichen Testsysteme kann immer nur genau eine Schnittstelle getestet werden, und es ist keine gleichzeitige Durchführung von unterschiedlichen Tests auf einem Testsystem möglich. Darüber hinaus sind die einzelnen Schnittstellentests nicht aufteilbar, sondern müssen immer am Stück durchgeführt werden.

2.2.3. Tester können nur jeweils eine Schnittstelle gleichzeitig testen

Es ist nicht möglich, dass ein Tester mehrere Schnittstellen auf unterschiedlichen Testsystemen zur gleichen Zeit testet, sondern jeder Tester beschäftigt sich für die

gesamte Zeit eines Schnittstellentests nur mit dem Test dieser einen Schnittstelle. Die Einhaltung dieser Restriktion ist oft schwierig, da ein Tester in der Regel für mehrere Schnittstellen zuständig ist (einzelne Tester sind für bis zu 50 Schnittstellen verantwortlich) und sichergestellt werden muss, dass nicht zwei von ihm zu testende Schnittstellen zur gleichen Zeit auf unterschiedlichen Testsystemen eingeplant werden.

2.2.4. Zeitrestriktionen für Tester

Manche Tester weisen zeitliche Restriktionen bezüglich ihrer Verfügbarkeit auf und können nicht beliebig für die Durchführung von Schnittstellentests eingeplant werden. Daher muss das zu erstellende Planungssystem berücksichtigen, dass diese Tester jeweils nur in den für sie zulässigen Zeitfenstern eingeplant werden.

2.2.5. Interdependenzen zwischen Testsystem und zu testender Komponente

Die einzelnen Schnittstellentests werden auf unterschiedlichen Testsystemen durchgeführt. Für einen Teil der Schnittstellentests können allerdings nur ausgewählte Testsysteme verwendet werden, da z. B. ein Teil der Softwarekomponenten hohe Anforderungen an die Leistungsfähigkeit der verwendeten Rechner stellen bzw. manche Komponenten auch auf leistungsschwächeren Testsystemen noch zuverlässig und mit akzeptabler Geschwindigkeit funktionieren müssen. Dies führt dazu, dass die zugehörigen Schnittstellen nicht auf beliebigen Testsystemen getestet werden können, sondern Interdependenzen zwischen Testsystem und installierter Komponente bestehen.

2.2.6. Inkompatibilitäten zwischen einzelnen Komponenten

Eine ähnliche Nebenbedingung liegt in der Unverträglichkeit von Komponenten. Im Produktportfolio des Softwareherstellers existieren unterschiedliche Produktlinien, welche aber teilweise zueinander inkompatibel sind. Daher können bestimmte Komponenten nicht gleichzeitig auf dem gleichen Testsystem installiert und getestet werden, sondern es muss durch das Planungssystem sichergestellt werden, dass Tests von zueinander inkompatiblen Komponenten auf jeweils unterschiedlichen Testsystemen durchgeführt werden.

2.2.7. Zeitliche Restriktionen

Ein automatisiertes Planungssystem muss bei der Aufstellung eines Zeitplans für die einzelnen Tester die vorgegebene Stundenanzahl pro Tag (üblicherweise acht), Mittagspausen sowie Wochenenden und Feiertage berücksichtigen.

3. Mögliche Lösungsansätze für das Planungsproblem

Aufgabe des zu erstellenden Planungssystems ist es, den menschlichen Planer bei der Erstellung eines Zeitplans für die Durchführung der Schnittstellentests auf den vorhandenen Testsystemen zu unterstützen. Hierbei ist die bloße Erstellung eines zulässigen Testplans, welcher die meisten der in Abschnitt 2.2 dargestellten Nebenbedingungen und Restriktionen berücksichtigt, trivial. Ein Testplan kann z.B. dadurch erzeugt werden, dass zufällig alle Schnittstellentests auf den einzelnen Testsystemen eingeplant werden. Anschließend werden iterativ bestehende Verstöße gegen die Nebenbedingungen aus Abschnitt 2.2.2 bis 2.2.7 aufgelöst und

Schnittstellentests, welche gegen bestimmte Restriktionen verstoßen, am Ende des Planungszeitraums eingeplant.

Als Problem einer derartigen Vorgehensweise stellt sich allerdings heraus, dass der so ermittelte Testplan sowohl gegen Restriktion 2.2.1 verstößt, als auch nicht die in Abschnitt 2.1 formulierten Planungsziele berücksichtigt. Bei einem derartigen Zeitplan existieren keine zusammenhängende Testzeiträume für die einzelnen Tester und auch der Gesamtzeitraum für die Durchführung der Tests ist inakzeptabel. Daher müssen in der Praxis einsetzbare und von den Testern akzeptierte Testpläne sowohl alle Restriktionen aus Abschnitt 2.2 erfüllen, als auch zusätzlich möglichst gut bezüglich der in 2.1 aufgeführten Zielfunktionen sein. Für die Ermittlung derartiger Pläne müssen Planungsverfahren entwickelt und eingesetzt werden, welche qualitativ hochwertige Testpläne generieren können.

Im Folgenden soll ein kurzer Überblick über mögliche Lösungsansätze für das Schnittstellenplanungsproblem gegeben werden und die unterschiedlichen Ansätze bezüglich ihrer Eignung zur Lösung des Problems betrachtet werden.

3.1. Gegenwärtiger Lösungsansatz

Beim Partnerunternehmen wurde bisher die Aufstellung eines Zeitplans für die Durchführung der Schnittstellentests durch einen menschlichen Planer mithilfe eines Tabellenkalkulationsprogramms (Excel) durchgeführt. Der Planungsprozess besteht aus der Ermittlung aller zu planenden Schnittstellen und der manuellen Einplanung aller Schnittstellentests auf die einzelnen Testsysteme. Der menschliche Planer geht hierbei so vor, dass er zu Beginn des Planungsprozesses Komponenten mit vielen Schnittstellen so am Stück eingeplant, dass die Restriktionen aus Abschnitt 2.2 berücksichtigt werden und darüber hinaus der entsprechende Tester möglichst am Stück testet. Anschließend plant er die anderen Schnittstellentests iterativ durch Versuch und Irrtum ein. Hierbei wird immer wieder manuell überprüft, ob nicht gegen eine vorgegebene Restriktion verstoßen wird und gleichzeitig versucht, die in Abschnitt 2.1 formulierten Planungsziele möglichst gut zu erreichen.

Durch den gegenwärtigen, manuellen Planungsvorgang können Testpläne ermittelt werden, welche von den Testern akzeptiert werden. Das Hauptproblem bei der gegenwärtigen Planungsmethode liegt in dem hohen Aufwand, welcher mit der Erstellung eines Testplans verbunden ist. Im Moment benötigt ein erfahrener Planer für die Aufstellung eines durchschnittlichen Testplans (ca. 300-500 zu planende Schnittstellen) ca. zwei bis drei Tage, wobei der resultierende Plan in der Regel immer gegen einige wenige Restriktionen aus Abschnitt 2.2 verstößt. Da gegenwärtig ca. 10 Testpläne pro Jahr erstellt werden müssen, ist der gesamte Aufwand in Personentagen alleine für die Erstellung der entsprechenden Testpläne recht hoch. Ziel des durchgeführten Projektes ist es, die Arbeit des menschlichen Planers durch ein automatisiertes Planungswerkzeug zu unterstützen, welches in ein entsprechendes EUS integriert werden soll.

3.2. Exakte Verfahren

In der Literatur, insbesondere im Operations Research und in der Informatik, wurde eine Reihe von unterschiedlichen exakten Planungsverfahren für Produktionsprobleme vorgestellt, welche nach entsprechenden Anpassungen auch für das hier vorliegende Schnittstellenplanungsproblem eingesetzt werden können [Zäpf82, Zäpf94,Sche91,DaWa97,KäTe04]. Das Schnittstellenplanungsproblem kann hier-

bei als eine Variante des Maschinenbelegungsproblems formuliert werden, welches ein Standardproblem der Produktionswirtschaft darstellt. Hierbei müssen Jobs (Schnittstellen) so auf unterschiedlichen Maschinen (Testsystemen) eingeplant werden, dass sowohl die Restriktionen eingehalten, als auch die vorgegebenen Zielfunktionen minimiert werden. Als Lösungsverfahren für derartige Probleme können z. B. vollständige Enumeration, Entscheidungsbaumsuchverfahren, begrenzte Enumeration oder Branch-and-Bound Ansätze verwendet werden. Das Hauptproblem all dieser Verfahren liegt in der hohen Komplexität und Größe des vorliegenden Schnittstellenplanungsproblems. Das Problem ist NP-vollständig [GaJo79], was bedeutet dass der Aufwand von exakten Verfahren bzw. Enumerationsverfahren exponentiell mit der Problemgröße (Anzahl Schnittstellen) ansteigt. Daher können exakte Verfahren das Problem nicht in akzeptabler Zeit lösen.

3.3. Lösungsheuristiken

Ein anderer Weg wird durch Lösungsheuristiken beschritten. Lösungsheuristiken (oft auch vereinfacht als Heuristiken bezeichnet) sind Verfahren, welche für das zu lösende Planungsproblem eine befriedigende Lösung finden sollen. Explizit wird hierbei auf die Sicherstellung der Ermittlung der optimalen Lösung verzichtet und sich mit einer qualitativ guten Lösung zufrieden gegeben. Heuristiken werden oft in Anlehnung an praktisch handhabbare oder plausibel erscheinende Vorgehensweisen so konstruiert, dass sie Planungsprobleme in relativ kurzer Zeit mit vertretbarer Lösungsqualität lösen. Das Hauptproblem beim Einsatz derartiger Verfahren liegt in der Ermittlung heuristischer Vorgehensweisen, welche in vernünftiger Zeit eine gute Lösung bezüglich vorgegebener Zielkriterien liefern (vgl. z. B. [Brü95]).

Ein möglicher Ansatzpunkt für die Ableitung einer geeigneten Lösungsheuristik für das Planungsproblem stellt das in Abschnitt 3.1 beschriebene, gegenwärtig eingesetzte, manuelle Planungsverfahren dar. Es wurde folglich im Rahmen der Entwicklung eines Planungssystems versucht, „Daumenregeln“ aus dem Einplanungsverhalten des menschlichen Planers zu extrahieren und diese dann als Heuristiken zu formulieren. Hierbei konnten zwar grundlegende Handlungsstrukturen des menschlichen Planers erkannt werden (z. B. „plane zuerst Komponenten mit vielen Schnittstellen am Stück ein“ oder „erzeuge zuerst zufälligen Plan und löse anschließend Restriktionsverletzungen so auf, dass alle unbelegten Zeitslots, welche größer sind als der zu verschiebende Zeitslot, überprüft werden und die entsprechenden Tests an die Stelle verschoben werden, welche am besten die Zielfunktionen aus Abschnitt 2.1 erfüllt“), es ist aber nicht gelungen, diese so zu beschreiben und miteinander zu kombinieren, dass eine daraus resultierende Heuristik zu zufrieden stellenden Ergebnissen geführt hätte.

Darüber hinaus wurden im Rahmen der Produktionssteuerung eine Vielzahl an statischen Prioritätsregeln entwickelt [GGR92, Kurb99], welche für die Reihenfolgeplanung von Arbeitsgängen im Rahmen der Fertigungsauftragsplanung eingesetzt werden können. Hierbei werden durch statische Prioritätsregeln Arbeitsaufträge (vergleichbar mit Schnittstellentests), welche in einer Wartschlange vor einzelnen Maschinen (Testsystemen) zur Bearbeitung anstehen, entsprechend ihrer Charakteristika und unabhängig vom aktuellen Stand des Produktionssystems eingeplant. Im Rahmen des vorliegenden Projekts wurde versucht, entsprechende Prioritätsregeln auf das Schnittstellenplanungsproblem anzupassen; es zeigte sich jedoch, dass die üblichen, bei statischen Prioritätsregeln verwendeten Kriterien (z. B. Zahl der auszuführenden Arbeitsgänge, Summe der benötigten

Bearbeitungszeiten oder festgelegte, spätestmögliche Fertigstellungstermine) nicht auf das Problem der Schnittstellenplanung übertragbar waren.

3.4. Metaheuristiken

Metaheuristiken (engl. „metaheuristics“) entstanden an der Schnittstelle zwischen Informatik und Künstlichen Intelligenz und stellen Methoden zur Verfügung, welche den Nutzer in die Lage versetzen auch komplexe Planungsprobleme in kurzer Zeit optimal (bzw. annähernd optimal) zu lösen. Derartige Methoden zeichnen sich dadurch aus, dass sie einfache, oft der Natur oder der Physik entlehnte Prinzipien („naturinspiriert") verwenden, wodurch eine zügige Modellierung und effiziente Lösung von Planungsproblemen ermöglicht wird.

Beispiele für Metaheuristiken sind lokale Suchverfahren (z. B. „Simulated Annealing" [LaAa88], „Tabu Search" oder „Stochastic Hill Climbing"), Evolutionäre Algorithmen [Gold89, Schw95] oder auch kombinierte Verfahren (z. B. COSA [Wend95]). Derartige Verfahren wurden in den letzten Jahren mit Erfolg in der Produktionsfeinplanung [KuRo95, KSS95, Kurb99, ScMe00, WaZä00] oder auch bei der Erstellung von Zeitplänen [AaLe97] sowie anderen Anwendungsbereichen (vgl. [BiNi95, Voss+99, Alan00]) eingesetzt.

Metaheuristiken lösen ein Planungsproblem dadurch, dass sie mit Hilfe von geeigneten Suchoperatoren iterativ neue, zufällige Lösungen generieren und sich bei der Suche durch den Lösungsraum auf gute Lösungen (bezüglich einer vorgegebenen Zielfunktion) fokussieren. Für den Einsatz von Metaheuristiken ist es notwendig, dass

1. vollständige Lösungen des Planungsproblems (z. B. komplette Zeitpläne) so kodiert werden, dass geeignete Suchoperatoren auf das Problem angewandt werden können, und
2. dass die vollständigen Lösungen bezüglich ihrer Qualität mithilfe einer vorgegebenen Zielfunktion bewertet werden können.

Metaheuristiken lassen sich bezüglich der eingesetzten Suchoperatoren, sowie der Anzahl der jeweils gleichzeitig betrachteten Lösungen charakterisieren. Eine grobe Charakterisierung unterscheidet zwischen lokalen und rekombinationsbasierten Verfahren. Bei den meisten lokalen Suchverfahren (z. B. „Simulated Annealing“, „Tabu Search“, „Adaptive Memory Programming“, „Random Walk“ oder „Iterated Local Search“) wird in jeder Iteration jeweils nur eine einzige Lösung betrachtet und im nächsten Suchschritt durch eine geringe Veränderung eine neue Lösung mit ähnlichen Eigenschaften erzeugt. Unterschiede zwischen den einzelnen Verfahren liegen in der Definition des entsprechenden Suchoperators und in der Steuerung der Suche. Im Gegensatz hierzu wird bei rekombinationsbasierten Metaheuristiken (z. B. Genetische Algorithmen, „Genetic Programming“, Evolutionsstrategien oder „Scatter Search“) üblicherweise eine Menge von Lösungen (Population) verwendet und neue Lösungen iterativ mit Hilfe von Rekombination erzeugt. Rekombinationsoperatoren erzeugen aus einer Menge von Ausgangslösungen neue Lösungen durch die Kombination von Teilen bzw. charakteristischen Eigenschaften der Ausgangslösungen. Im Gegensatz zu lokalen Suchverfahren, welche überwiegend im Bereich der Informatik und dem Operations Research weiterentwickelt wurden, entstammen rekombinationsbasierte Ansätze eher der Biologie und werden zu einem großen Teil im Anwendungsbereich weiterentwickelt.

Metaheuristiken zeichnen sich dadurch aus, dass sie universell einsetzbar sind und deswegen sehr einfach und flexibel auf unterschiedliche Problemstellungen ange-

passt werden können. Darüber hinaus sind die Anforderungen an die Struktur der zu lösenden Probleme sehr gering (es muss nur möglich sein, die Lösungsqualität von unterschiedlichen Lösungen zu bestimmen) und sie können (im Gegensatz z. B. zu exakten Verfahren wie der linearen Programmierung) auch für nichtlineare, nichtdifferenzierbare oder nichtstetige Probleme eingesetzt werden. Als Nachteile sind zu sehen, dass Standard-Metaheuristiken nur für kleine Probleme gute Ergebnisse liefern und größere Probleme nur durch speziell angepasste Verfahren, sowie einer problemadäquaten Auswahl und Einstellung der jeweiligen Methodenparameter gelöst werden können. Darüber hinaus ist aufgrund des zufälligen Charakters der Suche nicht sichergestellt, dass die optimale Lösung gefunden wird und der rechentechnische Aufwand für das Finden von guten Lösungen oft recht hoch (aber trotzdem wesentlich geringer als bei exakten Methoden).

4. Konzeption eines metaheuristikbasierten Planungssystems für die Durchführung von Schnittstellentests

Im Rahmen der Überlegungen zur Auswahl eines geeigneten Ansatzes für die Lösung des Schnittstellenplanungsproblems wurde ein genetischer Algorithmus (GA) ausgewählt, welcher zur Klasse der populationsbasierten Metaheuristiken gehört. Exakte Lösungsverfahren wurden nicht eingesetzt, da diese das vorliegende Planungsproblem unter Berücksichtigung der vorhandenen Nebenbedingung nicht in akzeptabler Zeit lösen können. Weitere Probleme für exakte Verfahren wurden in der NP-Vollständigkeit des Planungsproblems gesehen, welche dazu führt, dass der Aufwand zum Lösen des Problems exponentiell mit der Problemgröße steigt. Darüber hinaus wurde im Rahmen von konzeptionellen Vorstudien versucht geeignete Heuristiken zur Lösung des Problems zu entwickeln (vgl. Abschnitt 3.3). Die praktische Anwendung zeigte jedoch, dass entweder die Qualität der damit erzeugten Testpläne unzureichend war oder nicht sichergestellt werden konnte, dass alle in Abschnitt 2.2 formulierten Nebenbedingungen ausreichend erfüllt wurden. Die folgenden Abschnitte beschreiben das Fachkonzept des realisierten metaheuristikbasierten Planungssystems.

4.1. Testplankodierung und Testplanerzeugung

Wie in Abschnitt 3.4 beschrieben, muss beim Einsatz von Metaheuristiken ein gültiger und vollständiger Testplan so kodiert werden, dass Suchoperatoren darauf angewendet werden können (vgl. [Roth02]). Dies bedeutet, dass vollständige Lösungen Informationen darüber enthalten müssen, welche Schnittstellentests zu welchen Zeiten auf welchen Testsystemen durchgeführt werden. Da jeder Softwarekomponente genau ein Tester zugeordnet ist, muss keine zusätzliche Zuordnung zwischen Schnittstelle und Ansprechpartner (Tester) vorgenommen werden und es ist ausreichend nur die jeweilige Schnittstelle bei der Planung zu betrachten. Für das vorliegende Schnittstellenplanungsproblem wurde eine gewichtete Kodierung [Bean92] zur Kodierung eines Zeitplans eingesetzt. Gewichtete Kodierungen können für Permutationsprobleme verwendet werden und wurden bisher für Maschinenbelegungs-, Fahrzeugrouting-, Kapazitätszuordnungs-, Netzwerk- und „Travelling-Salesperson"-Probleme verwendet. Für einen Überblick über den Einsatz von gewichteten Kodierungen vergleiche [Norm95, RaJu00, Roth02].

Beim Einsatz einer gewichteten Kodierung wird durch einen Vektor r aus n zufälligen reellwertigen Zahlen $r_i \in [0,k[$ mit $r_i \neq r_j \forall i \neq j$ beschrieben, in welcher Reihenfolge die n Schnittstellen auf den k verschiedenen Testsystemen eingeplant werden. Hierbei wird jede einzuplanende Schnittstelle mit einer eindeutigen Nummer $i \in \{0,1,...,n-1\}$ und jedes Testsystem mit einer eindeutigen Nummer $j \in \{0,1,...,k-1\}$ versehen. Die zeitliche Abfolge der Tests wird durch die Position und die relativen Werte der einzelnen Elemente r_i des Vektors bestimmt. Darüber hinaus wird bei der Erzeugung des Testplans Schnittstelle i jeweils auf dem Testsystem $j=\lfloor r_i \rfloor$ eingeplant. Die Reihenfolge, in der die jeweiligen Tests durchgeführt werden, wird durch die Positionen der einzelnen Elemente des Vektors entsprechend ihres Wertes in absteigender Ordnung festgelegt.

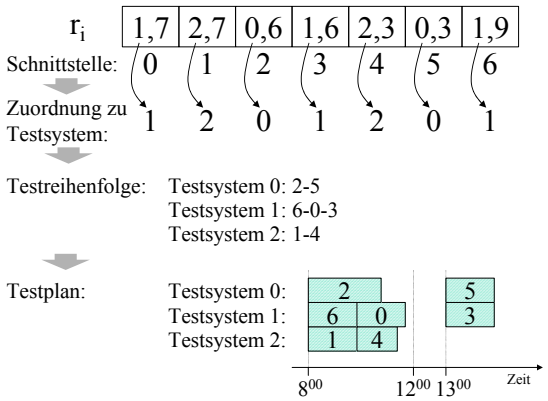

Abbildung 1: Erzeugung eines Zeitplans aus einer kodierten Lösung

Ein mögliches Beispiel für einen gewichteten Vektor stellt r=(1,7; 2,7; 0,6; 1,6; 2,3; 0,3; 1,9) mit n=7 und k=3 dar (vgl. Abbildung 1). Es werden also sieben Schnittstellen (Vektor hat sieben Elemente) auf drei Testsystemen (die drei möglichen „Vorkommaziffern" sind 0, 1 oder 2) eingeplant. Im Zeitplan werden daher die Schnittstellen mit der Nummer 2 (r_2=0,6) und 5 (r_5=0,3) auf Testsystem 0 in der Reihenfolge 2-5 (r_2>r_5) eingeplant. Auf Testsystem 1 werden die Schnittstellen 0 (r_0=1,7), 3 (r_3=1,6) und 6 (r_6=1,9) in der Reihenfolge 6-0-3 (r_6>r_0>r_3) getestet. Weiterhin werden auf Testsystem 2 die Schnittstellen 1 (r_1=2,7) und 4 (r_4=2,3) in der Reihenfolge 1-4 (r_1>r4) eingeplant.

Entsprechend dieser Vorgehensweise kann aus jedem gewichteten Vektor r der Länge n mit $r_i \in [0,k[$ eine zulässige Reihenfolge der Schnittstellentests auf den k verschiedenen Testsystemen gewonnen werden. Durch eine derartige Kodierung wird ein Teil der in Abschnitt 2.2 formulierten Nebenbedingungen berücksichtigt. Da bei der Verwendung einer gewichteten Kodierung jede Schnittstelle nur ein einziges Mal eingeplant werden kann, wird sichergestellt, dass jeder Schnittstellentest genau nur ein einziges Mal durchgeführt wird. Darüber hinaus kann auf einer Maschine zum gleichen Zeitpunkt auch nie mehr als ein Test durchgeführt werden, sodass automatisch die Nebenbedingung aus Abschnitt 2.2.2 eingehalten wird.

Nachdem durch die Kodierung festgelegt worden ist, in welcher Reihenfolge die Schnittstellen auf den Testsystemen eingeplant werden, muss in einem zweiten Schritt aus der Reihenfolge ein Zeitplan erstellt werden. Bei dessen Erstellung werden die zeitlichen Restriktionen aus Abschnitt 2.2.7 wie z. B. maximale tägli-

che Arbeitszeit, Mittagspause oder Wochenenden berücksichtigt (vgl. Abbildung 1). Am Ende des Konstruktionsprozesses steht ein Zeitplan für die Durchführung der Tests, welcher die Nebenbedingungen 2.2.2 und 2.2.7 berücksichtigt.

4.2. Testplanbewertung

Beim Einsatz von Metaheuristiken muss die Qualität eines Testplans mithilfe einer Bewertungsfunktion bewertet werden. Aufgrund der vorhandenen unterschiedlichen Ziele (vgl. Abschnitt 2.1) liegt ein multikriterielles Problem vor, bei dem paretooptimale Lösungen bezüglich der vorgegebenen Planungsziele ermittelt werden können. Prinzipiell sind populationsbasierte Metaheuristiken gut für die Lösung von derartigen multikriteriellen Problemen geeignet [Deb01], allerdings ist ein größerer Aufwand für die Ermittlung von guten Lösungen notwendig. Da für die Lösung des vorliegenden Planungsproblems keine Menge von paretooptimalen Lösungen benötigt wird, wird daher eine additive Bewertungsfunktion verwendet, bei der ein Zeitplan einzeln bezüglich der vorgegebenen Ziele und Restriktionen bewertet wird und die sich daraus ergebenden Teilbewertungen additiv zusammengefasst werden.

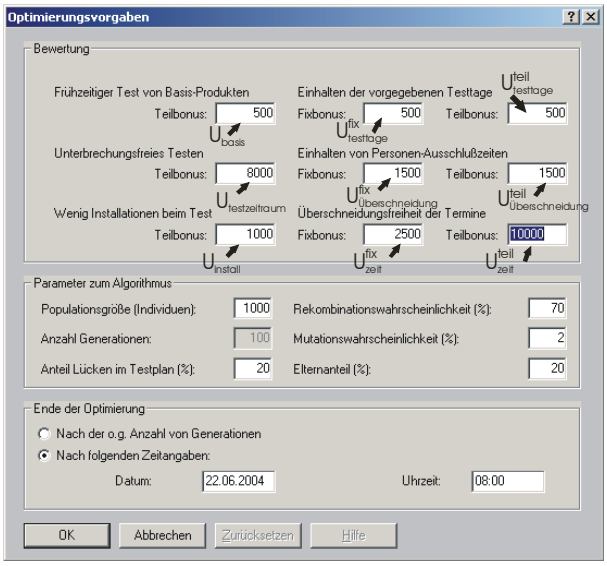

Abbildung 2: Gewichtungen für Zeitplanbewertung und Parameter für GA

Die Bewertung eines Zeitplans setzt sich somit aus Teilbewertungen zusammen, welche vom Erreichen der unterschiedlichen Planungsziele aus Abschnitt 2.1 und dem Grad der Einhaltung der Nebenbedingungen aus Abschnitt 2.2 abhängen. Die jeweiligen Gewichtungen U für die einzelnen Ziele und Restriktionen werden durch den Entscheider vorgegeben (vgl. Abbildung 2) und beeinflussen die Qualität der durch die Metaheuristik (GA) erzeugten Zeitpläne. Je größer die jeweilige Gewichtung gewählt wird, desto stärker wird dieses Ziel bzw. die Einhaltung einer Restriktion durch das metaheuristikbasierte Planungssystem berücksichtigt. Die Bewertung eines Zeitplans hängt damit analog zu Abschnitt 2 von den folgenden Kriterien ab:

- **Zusammenhängende Testzeiträume/Unterbrechungsfreies Testen**: Die Bewertung eines Zeitplans wird um $B_{testzeitraum}=U_{testzeitraum}(l_{testzeitraum}/n)$ erhöht, wobei $l_{testzeitraum}$ die kumulierte Anzahl der Schnittstellen bezeichnet,

welche jeweils ohne Unterbrechung durch einen Tester getestet werden können, und n die Gesamtanzahl der einzuplanenden Schnittstellen angibt. $U_{testzeitraum}$ ist eine Gewichtung, welche durch den Entscheider vorgegeben wird (vgl. Abbildung 2).

- **Frühzeitiger Test von Basiskomponenten**: Die Bewertung eines Zeitplans wird um die Teilbewertung $B_{basis} = U_{basis}(1/l_{basis} \sum_{Basiskomp.}(1-(pos_i/n))$ erhöht, wobei l_{basis} die Anzahl der Basiskomponenten und pos_i die Position der Basiskomponente i im Zeitplan bezeichnet.
- **Minimierung der Installationszeiten**: Die Bewertung eines Zeitplans wird um $B_{install} = U_{install}(m_{install}/(n_{install}k))$ erhöht, wobei $m_{install}$ die Anzahl der im Zeitplan geplanten Installationen, $n_{install}$ die Anzahl der notwendigen Installationen und k die Anzahl der Testsysteme bezeichnet.

Analog zu den Zielen wird der Grad der Nichtberücksichtigung von Nebenbedingungen ebenfalls bei der Planbewertung berücksichtigt:

- **Anzahl benötigter Testtage**: Falls die Anzahl der vorgegebenen Testtage eingehalten wird, wird die Bewertung eines Zeitplans um $B_{testtage}=U^{fix}_{testtage}$ erhöht. Ansonsten gilt $B_{testtage}=U^{teil}_{testtage}(2-(t-t_{min}))$, wobei t_{min} die minimal mögliche Anzahl an Testtagen und t die aus dem Zeitplan resultierende Anzahl an Testtagen bezeichnet.
- **Gleichzeitiger Test**: Falls keine Überschneidungen im Zeitplan auftreten, wird die Bewertung eines Zeitplans um $B_{Überschneidung} = U^{fix}_{Überschneidung}$ erhöht. Ansonsten gilt $B_{Überschneidung} = U^{teil}_{Überschneidung}(1-(n_ü/n))$, wobei $n_ü$ die Gesamtanzahl der Überschneidungen im Zeitplan bezeichnet.
- **Zeitrestriktionen für Tester**: Falls keine Zeitrestriktionen verletzt werden, gilt $B_{Zeit} = U^{fix}_{Zeit}$. Ansonsten ergibt sich $B_{Zeit} = U^{teil}_{Zeit}(1-(n_z/m_z))$, wobei n_z die Anzahl der Schnittstellen im Zeitplan bezeichnet, bei denen Zeitrestriktionen verletzt werden, und m_z die Gesamtanzahl der Schnittstellen darstellt, an denen Tester mit Zeitrestriktionen beteiligt sind.

Die Gesamtbewertung für einen Zeitplan berechnet sich somit als $B_{gesamt} = B_{testzeitraum} + B_{basis} + B_{install} + B_{testtage} + B_{Überschneidung} + B_{Zeit}$. Durch die Wahl der jeweiligen Gewichtungen U werden die einzelnen Ziele und Restriktionen unterschiedlich stark gewichtet und dementsprechend mögliche Lösungen (Zeitpläne) bei der Suche durch die Metaheuristik unterschiedlich gut bewertet.

4.3. Initialisierung von Startlösungen

Bei der Verwendung von Metaheuristiken, bzw. GA, werden die Startlösungen üblicherweise zufällig erzeugt. Auch im vorliegenden Fall wird eine Menge von Zeitplänen dadurch zufällig erzeugt, dass unterschiedliche Vektoren r mit jeweils zufälligen Elementen $r_i \in [0,k[$ mit $r_i \neq r_j \forall i \neq j$ ermittelt werden. Darüber hinaus können bei einer geschickten Initialisierung der Startlösungen sowohl vor-

handenes Vorwissen über Eigenschaften von qualitativ guten Lösungen, als auch bestehende Nebenbedingungen berücksichtigen werden. Von beiden Möglichkeiten wurde bei der Lösung des Schnittstellenplanungsproblems mit Hilfe von GA Gebrauch gemacht.

Bei der Untersuchung des Verhaltens von menschlichen Planern (vgl. Abschnitt 3.1) hat sich gezeigt, dass bei guten Lösungen Tester immer eine große Menge an Schnittstellen ohne Unterbrechungen testen und nur geringe Lücken im Zeitplan auftreten. Dieses Vorwissen über die Struktur von guten Zeitplänen wurde bei der Initialisierung der Startpopulation berücksichtigt und Tester mit vielen Schnittstellen jeweils ohne zeitliche Unterbrechungen im Zeitplan eingeplant. Dies wurde so realisiert, dass nach der zufälligen Initialisierung alle Schnittstellen ermittelt wurden, welche durch einen bestimmten Tester getestet werden. Anschließend wurden die den einzelnen Schnittstellen zugeordneten Elemente r_i im kodierten Zeitplan so gewählt, dass diese untereinander nur geringe Unterschiede aufweisen (jeweils nur um 0.001 unterschiedlich). Dadurch, dass alle Schnittstellen eines Testers sehr ähnliche Gewichte r_i im Vektor r aufweisen, werden sie bei der Testplanerzeugung nacheinander im Zeitplan eingeplant und der entsprechende Tester kann ohne Unterbrechungen testen. Entsprechend dieser Vorgehensweise wurden alle Schnittstellen der zehn Tester mit der größten Anzahl an zu testenden Schnittstellen modifiziert. Da allerdings jeder Schnittstelle zwei verschiedene Tester zugeordnet sind, kann dieses Vorgehen nur für eine kleine Anzahl an Testern (jeweils mit der höchsten Anzahl an zu testenden Schnittstellen) eingesetzt werden und darüber hinaus können bei diesem Vorgehen auch schon modifizierte Schnittstellen abermals verändert werden.

Neben der Berücksichtigung von Vorwissen über die Struktur von guten Lösungen wird bei der Initialisierung der Startpopulation auch von der Möglichkeit Gebrauch gemacht, die Nebenbedingungen aus Abschnitt 2.2.5 und 2.2.6 zu berücksichtigen. Zur Erfüllung dieser Nebenbedingungen ist es notwendig, dass manche Schnittstellen nur auf bestimmten Testsystemen getestet werden. Daher werden nach dem zufälligen Erzeugen der Startlösungen und der stückweisen Einplanung von Schnittstellen (vgl. vorherigen Absatz) in einem nächsten Schritt alle Schnittstellen betrachtet, welche Nebenbedingungen aus Abschnitt 2.2.5 oder 2.2.6 verletzen. Bei allen derartigen Schnittstellen i wird das entsprechende Element r_i des Lösungsvektors so abgeändert (die Vorkommastelle modifiziert), dass keine Verstöße gegen die zwei Nebenbedingungen mehr auftreten.

Ursprünglich ist im Rahmen des Projektes davon ausgegangen worden, dass es möglich ist, einen lückenlosen Testplan zu finden, bei dem keine Verletzungen von Nebenbedingungen auftreten. Es hat sich allerdings bei der Realisierung des Planungssystems schnell gezeigt, dass die Anzahl der Restriktionen so hoch ist, dass ein lückenloser Testplan nicht möglich ist, sondern - ähnlich wie bei manuell erstellen Testplänen - zusätzliche Lücken im Testplan eingeführt werden müssen. Daher wurde zu den schon vorhandenen Schnittstellen zusätzliche „Leerschnittstellen" hinzugefügt (üblicherweise ca. 15-20%), welche analog zu den real existierenden Schnittstellen behandelt werden, aber keine Restriktionen verletzen und auch keinen Ansprechpartner besitzen. Dadurch erhöht sich der Anzahl der zu planenden Schnittstellen (real existierende Schnittstellen plus Lücken im Testplan) und die Länge n der kodierten Lösungen muss entsprechend erhöht werden.

4.4. Suchoperatoren und Steuerung der Suche

Als Metaheuristik zur Steuerung der Suche wurde ein Genetischer Algorithmus [Gold89] gewählt. GA wenden Suchoperatoren (Rekombination und Mutation) auf eine Menge (Population) von Problemlösungen (Individuen) über mehrere Iterationen (Generationen) an. GA wurden schon mit Erfolg für ähnliche Planungsprobleme eingesetzt [KSS95, KuRo95, WaZä00], sind einfach anwendbar und verwenden eine Population von Lösungen. Das prinzipielle Ablaufschema eines GA ist in Abbildung 3 dargestellt. Als Abbruchkriterium wurde eine maximale Anzahl an Iterationen bzw. eine maximal zur Verfügung stehende Rechenzeit für den GA gewählt (vgl. hierzu Abbildung 2).

Initialisiere Population von Lösungen
- Erzeuge Startlösungen
- Bewerte Startlösungen

Solange Abbruchkriterium nicht erfüllt
- Wähle gute Lösungen mit Hilfe eines Selektionsverfahrens aus
- Erzeuge neue Lösungen aus den ausgewählten Lösungen mit Hilfe von 1-Punkt Crossover
- Mutiere neue Lösungen
- Bewerte neue Lösungen

Abbildung 3: Ablaufschema eines Genetischen Algorithmus

Als Rekombinationsverfahren wurde 1-Punkt Crossover gewählt. Hierbei werden mit der Wahrscheinlichkeit P_{cross} zwei Lösungen zufällig ausgewählt und zwei neue Lösungen dadurch erzeugt, dass ein so genannter Crossover-Punkt bestimmt wird und der Teil der Lösung 1 vor dem Crossover-Punkt mit dem Teil des anderen Individuums (Lösung 2) nach dem Crossover-Punkt kombiniert wird (vgl. Abbildung 4). Als Mutationsoperator wird eine zufällige Veränderung von r_i durchgeführt. Das bedeutet also, dass die Einplanung (sowohl Testsystem, als auch Reihenfolge) der Schnittstelle i zufällig verändert wird. Bei der Durchführung einer Mutation wird allerdings sichergestellt, dass die Restriktionen aus Abschnitt 2.2.5 und 2.2.6 berücksichtigt werden. Für den Ablauf des GA wurde als Wahrscheinlichkeit für Crossover $P_{cross} \approx 0{,}8$ und für Mutation $P_{mut} \approx 2/n$ gewählt. Als Selektionsverfahren wurde eine $\mu + \lambda$-Strategie [Bäck98] gewählt (üblicherweise $\mu = 200$ und $\lambda = 800$). Dies bedeutet, dass aus μ Ausgangslösungen jeweils λ neue Lösungen erzeugt werden. Anschließend werden aus den $\mu + \lambda$ Lösungen dann wieder die μ besten Lösungen ausgewählt.

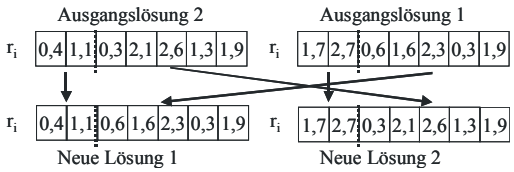

Abbildung 4: 1-Punkt Crossover

5. Einsatz und Praxiserfahrungen

Das im vorherigen Abschnitt beschriebene Konzept eines Planungssystems wurde in enger Zusammenarbeit mit der Qualitätssicherungsabteilung konzipiert und vollständig innerhalb des Unternehmens mit Hilfe von Visual Basic implementiert. Visual Basic wurde aus Gründen der Homogenität zu den schon vorhandenen Anwendungen in der Qualitätssicherung eingesetzt. Abbildung 5 zeigt den Hauptdialog der entstandenen Anwendung.

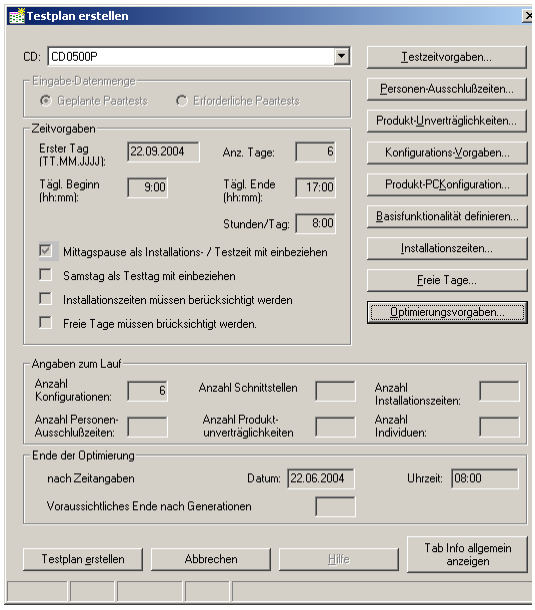

Abbildung 5: Hauptdialog des Planungswerkzeuges

Neben dem eigentlichen Planungssystem zur Ermittlung geeigneter Zeitpläne mithilfe eines GA wurden zusätzliche Funktionalitäten im Rahmen des EUS realisiert. So werden die für die Planung jeweils benötigten Daten automatisch aus den entsprechenden Datenbanken (MS-Access) ausgelesen, dargestellt und verwaltet. Weiterhin kann der durch das metaheuristikbasierte Planungssystem ermittelte Testplan durch den Entscheider noch manuell verändert werden und basierend auf dem endgültigen Testplan sowohl automatisch Einladungen an die jeweiligen Tester verschickt, als auch Testpläne für die Durchführung der Tests ausgedruckt werden.

Aufgrund der Komplexität des Planungsproblems und der großen Menge an Nebenbedingungen sind die Rechenzeiten für die Ermittlung von guten Zeitplänen hoch. Im praktischen Einsatz werden bei einer bevorstehenden Qualitätssicherungsphase üblicherweise zwei Planungsläufe auf zwei unterschiedlichen Arbeitsplatzrechnern über Nacht (ca. 8-10 Std.) durchgeführt. Die während dieser Zeit durch den GA ermittelten Zeitpläne entsprechen den Anforderungen des Planers und es werden in den allermeisten Fällen nur minimale Änderungen am so ermittelten Zeitplan vorgenommen. Ursprünglich wurde geplant, Laufzeitverbesserungen durch genauere Untersuchungen der jeweils am besten geeigneten GA-Parameter zu versuchen und darüber hinaus die Implementierung (Visual Basic mit vielen Datenbankzugriffen) der Verfahren effizienter zu gestalten. Da allerdings die gegenwärtigen Rechenzeiten sowohl aus der Sicht des Unternehmens als auch aus Sicht der Qualitätssicherungsabteilung unproblematisch sind, wurde darauf verzichtet.

6. Zusammenfassung

Bei der Erstellung von komponentenbasierter Unternehmenssoftware muss im Rahmen der Qualitätssicherung nicht nur die Funktionalität der einzelnen Komponenten, sondern insbesondere auch das korrekte Zusammenspiel der unterschiedlichen Teilkomponenten überprüft werden. Hierzu müssen Schnittstellentests an zentraler Stelle durchgeführt und ein entsprechender Zeitplan für die Durchführung der Schnittstellentests aufgestellt werden. Das Aufstellen eines derartigen Zeitplans ist aufgrund der Komplexität des Planungsproblems und einer Vielzahl von Zielkriterien und Restriktionen sehr aufwendig und wurde bisher durch einen menschlichen Planer durchgeführt.

Das Ziel des Beitrags ist die Beschreibung der Konzeption und Umsetzung eines Entscheidungsunterstützungssystems, welches den menschlichen Planer bei der Planung von Schnittstellentests in der Qualitätssicherung unterstützt. Wichtigstes Element des EUS ist ein automatisiertes Planungssystem, welches Zeitpläne für die Durchführung der Schnittstellentests erzeugt. Im Rahmen des Beitrags wird zuerst eine genaue Beschreibung des zu lösenden Planungsproblems vorgenommen. Es ist ein Zeitplan für die Durchführung von Schnittstellentests zu ermitteln, welcher festlegt wann und auf welchen Testsystemen die einzelnen Schnittstellen getestet werden sollen. Es wird aufgezeigt, welche Planungsziele existieren und welche Nebenbedingungen dabei berücksichtigt werden müssen. Nach der Diskussion von möglichen Lösungsansätzen für das Problem wird ein metaheuristikbasiertes Planungssystem zur Erstellung von Zeitplänen entwickelt. Dieses wurde in enger Zusammenarbeit mit dem Partnerunternehmen konzipiert und umgesetzt. Bei der Konzeption des Planungsverfahrens sind die kritischen Erfolgsfaktoren die Art und Weise, wie Zeitpläne kodiert und erzeugt werden, wie die Qualität von unterschiedlichen Testplänen bewertet wird und wie Startlösungen für die Metaheuristik erzeugt werden. Als Metaheuristik für die automatisierte Zeitplanerzeugung wird ein Genetischer Algorithmus eingesetzt. Das resultierende Planungssystem wurde in ein EUS mit zusätzlichen Funktionalitäten zur Verwaltung und Durchführung von Schnittstellentests integriert und wird im Rahmen der Qualitätssicherung mit Erfolg eingesetzt.

Literatur

[AaLE97] Aarts, E., Lenstra, J. K.. Local Search in Combinatorial Optimization. John Wiley & Sons; New York, 1997.

[Alan00] Alander, J. T.: Indexed bibliography of genetic algorithms in economics. University of Vaasa, Department of Information Technology and Production Economics}, 94-1-ECO, 2000.

[Bäck98] Bäck, T.: An Overview of Parameter Control Methods by Self-Adaptation in Evolutionary Algorithms. Fundamenta Informaticae 35, 1998, S. 51-66.

[Bean92] Bean, J. C.: Genetics and random keys for sequencing and optimization (Technical Report 92-43). Ann Arbor, MI: Department of Industrial and Operations Engineering, University of Michigan, 1992.

[BiNi95] Biethahn, J.; Nissen, V.: Evolutionary Algorithms in Management Applications. Springer-Verlag, Berlin, 1995.

[Brü95] Brüggemann, W.: Ausgewählte Probleme der Produktionsplanung: Modellierung, Komplexität und neuere Lösungsmöglichkeiten. Physica, Heidelberg, 1995.

[DaWa97] Dangelmaier, W.; Warnecke, H.-J.: Fertigungslenkung: Planung und Steuerung des Ablaufs der diskreten Fertigung. Springer, Berlin, 1997.

[Deb01] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester, UK,2001.

[GaJo79] Garey, M. R.; Johnson, D. S.: Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman, New York, 1979.

[GGR92] Glaser, H.; Geiger, W.; Rohde, V.: PPS – Produktionsplanung und –Steuerung: Grundlagen-Konzepte-Anwendungen, Wiesbaden, 1992.

[Gold89] Goldberg, D. E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, 1989.

[KäTe04] Käschel, J.; Teich, T.: Produktionswirtschaft Band 1: Grundlagen, Produktionsplanung und –steuerung. GUC Verlag, Chemnitz, 2004

[KSS95] Kurbel, K.; Schneider, B.; Singh, K.: Parallelization of Hybrid Simulated Annealing and Genetic Algorithm for Short-term Production Scheduling. In Zhong, E. (Hrsg.): Proc. of the Int. Symp. on Intelligence, Knowledge and Integration for Manufacturing, S. 321-326, Nanjing, China, 1995.

[KuRo95] Kurbel, K.; Rohmann, T.: Ein Vergleich von Verfahren zur Maschinenbelegungsplanung: Simulated Annealing, Genetische Algorithmen und mathematische Optimierung, Wirtschaftsinformatik 36(6), 1995, S. 581-593.

[Kurb99] Kurbel, K.: Produktionsplanung und -steuerung - Methodische Grundlagen von PPS-Systemen und Erweiterungen, Oldenbourg-Verlag, München, 1999.

[LaAa88] van Laarhoven, P. J. M.; Aarts, E. H. L.: Simulated Annealing: Theory and Applications. Dordrecht, Kluwer, 1988.

[Norm95] Norman, B. A.: Scheduling using the random keys genetic algorithm. Unveröffentlichte Dissertation, University of Michigan. Ann Arbor, Michigan, 1995.

[RaJu00] Raidl, G. R.; Julstrom, B. A.: A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In Proceedings of the 2000 ACM Symposium on Applied Computing. ACM Press, 2000, S. 440–445.

[Roth02] Rothlauf, F.: Representations for genetic and evolutionary algorithms. Heidelberg, Springer, 2002.

[Sche91] Scheer, A.-W.: Fertigungssteuerung: Expertenwissen für die Praxis. R. Oldenbourg, München, 1991.

[Schw95] Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley&Sons, New York, 1995.

[SWM95] Schultz, J.; Weigelt, M.; Mertens, P.: Verfahren für die rechnergestützte Produktionsfeinplanung – ein Überblick, Wirtschaftsinformatik 37(6), 1995, S. 594-608.

[ScMe00] Schultz, J.; Mertens, P.: Untersuchung wissensbasierter und weiterer ausgewählter Ansätze zur Unterstützung der Produktionsfeinplanung - ein Methodenvergleich, Wirtschaftsinformatik 42(1), 2000, S. 56-65.

[Voss+99] Voß, S.; Martello, S.; Osman, I. H.; Roucairol C.: Advances and Trends in Local Search Paradigms for Optimization, Kluwer, Boston, 1999.

[WaZä00] Wasner, M.; Zäpfel,G.: A heuristic solution concept for a generalized machine sequencing problem with an application to radiator manufactoring, International Journal of Production Economics 68, 2000, S. 199-213.

[Wend95] Wendt, O.: Tourenplanung durch Einsatz naturanaloger Verfahren. Promotion. Wiesbaden, Gabler, 1995.

[Zäpf82] Zäpfel, G.: Produktionswirtschaft: operatives Produktionsmanagement. De Gruyter, Berlin, 1982.

[Zäpf94] Zäpfel, G.: Entwicklungsstand und –tendenzen in PPS-Systemen. In: Handbuch Produktionsmanagement. Hrsg.: Corsten, H., Gabler, Wiesbaden, 1994, S. 719-745.