

February 2007

# Konzeption und Implementierung eines Planungssystems für Schnittstellentests

Franz Rothlauf

*Universität Mannheim*, rothlauf@uni-mannheim.de

Follow this and additional works at: <http://aisel.aisnet.org/wi2007>

---

## Recommended Citation

Rothlauf, Franz, "Konzeption und Implementierung eines Planungssystems für Schnittstellentests" (2007). *Wirtschaftsinformatik Proceedings 2007*. 84.

<http://aisel.aisnet.org/wi2007/84>

This material is brought to you by the Wirtschaftsinformatik at AIS Electronic Library (AISeL). It has been accepted for inclusion in Wirtschaftsinformatik Proceedings 2007 by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact [elibrary@aisnet.org](mailto:elibrary@aisnet.org).

In: Oberweis, Andreas, u.a. (Hg.) 2007. *eOrganisation: Service-, Prozess-, Market-Engineering*; 8. Internationale Tagung Wirtschaftsinformatik 2007. Karlsruhe: Universitätsverlag Karlsruhe

ISBN: 978-3-86644-094-4 (Band 1)

ISBN: 978-3-86644-095-1 (Band 2)

ISBN: 978-3-86644-093-7 (set)

© Universitätsverlag Karlsruhe 2007

# Konzeption und Implementierung eines Planungssystems für Schnittstellentests

Franz Rothlauf

Lehrstuhl für ABWL und Wirtschaftsinformatik, Universität Mannheim  
rothlauf@uni-mannheim.de

## Abstract

Bei der Erstellung von komponentenbasierten, betrieblichen Informationssystemen 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 Softwareherstellers 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.

## 1 Einleitung

Bei der Erstellung von komponentenbasierten, betrieblichen Informationssystemen 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 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 Zeit-

plan für die Durchführung der Schnittstellentests erstellt werden muss. Das Aufstellen eines 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 gehört zur Klasse der resource-constrained project scheduling Probleme [Bruc99] und ist vergleichbar mit Maschinenbelegungs- oder Stundenplanbelegungsproblemen. Derartige Probleme gehören zu den traditionellen Forschungsgebieten der Betriebswirtschaftslehre und es existieren eine Vielzahl von exakten Methoden, welche hierfür optimale Lösungen ermitteln können [Brü95]. Aufgrund der Komplexität der zugrundeliegenden Planungsprobleme sind diese allerdings auf viele realistische Problemstellungen oft nicht anwendbar, so dass in zunehmenden Maße Metaheuristiken zur Lösung komplexer Planungsprobleme eingesetzt werden [Zäpf94, KäTe04]. Mit deren Hilfe können auch manche Planungsprobleme gelöst werden, welche klassischen, exakten Methoden nicht zugänglich sind [SWM95, ScMe00].

Der vorliegende Beitrag stellt die Konzeption und Umsetzung eines EUS für die Planung von Schnittstellentests vor. Das EUS wurde in Zusammenarbeit mit einem großen deutschen Softwarehersteller im Bereich Steuer- und Buchführungssoftware konzipiert und implementiert und wird mittlerweile mit Erfolg in der Qualitätssicherung eingesetzt. Das im EUS integrierte Planungssystem erstellt mit Hilfe einer Metaheuristik Zeitpläne für die Durchführung von Schnittstellentests und berücksichtigt 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.

## **2 Erstellung von Zeitplänen für Schnittstellentests**

Bei der komponentenbasierten Erstellung von Informationssystemen wird die Funktionalität der zu erstellenden Software in einzelne Komponenten zerlegt. Wichtig bei der Entwicklung komponentenbasierter Software ist, dass verbindliche und klar definierte Schnittstellen zwischen den Softwarekomponenten festgelegt werden. Es ist Aufgabe der Qualitätssicherung die Einhaltung der entsprechenden Schnittstellenspezifikationen zu überprüfen und Fehler im Zusammenspiel unterschiedlicher Komponenten aufzudecken.

Bei der vorliegenden Problemstellung wird der Schnittstellentest 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 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 Schnittstellen (üblicherweise 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.

## **2.1. Ziele des Planungsprozesses**

Die folgenden konfligierenden Zielsetzungen wurden im Rahmen des Projektes in Zusammenarbeit mit der Qualitätssicherungsabteilung ausgearbeitet.

**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 durchgeführt werden, aber es sollten keine Lücken im Testplan des jeweiligen Ansprechpartners entstehen.

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

Bei der Durchführung der Tests sollen Basiskomponenten vor Zusatzkomponenten getestet werden. Falls Basiskomponenten vorher getestet werden, kann eindeutig bestimmt werden, welche Fehler auf Basiskomponenten und welche auf Zusatzkomponenten zurückzuführen sind.

**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 die schon installierten Komponentenversionen verwendet. In der Regel müssen allerdings bei der Aufstellung eines Zeitplans zusätzliche Installationszeiten für Komponenten berücksichtigt werden. Beim erstmaligen Test einer Schnittstelle auf einem der vorhandenen Testsysteme muss dann die entsprechende Komponente 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 wird, dass die entsprechenden Komponenten auf einer möglichst geringen Anzahl an Testsystemen installiert werden.

## **2.2. Bei der Planung einzuhaltende Nebenbedingungen**

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

**(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) 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.

**(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 Dauer eines Tests nur mit dieser einen Schnittstelle. Die Einhaltung dieser Restriktion ist schwierig, da ein Tester i.d.R. für mehrere

Schnittstellen zuständig ist (einzelne Tester sind für bis zu 50 Schnittstellen zuständig) und sichergestellt werden muss, dass nicht zwei von ihm zu testende Schnittstellen zur gleichen Zeit auf unterschiedlichen Testsystemen eingeplant werden.

**(4) Zeitrestriktionen für Tester:** Manche Tester weisen zeitliche Restriktionen bezüglich ihrer Verfügbarkeit auf und können nicht beliebig für Schnittstellentests eingeplant werden.

**(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.

**(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 Tests von zueinander inkompatiblen Komponenten müssen auf jeweils unterschiedlichen Testsystemen durchgeführt werden.

**(7) Zeitliche Restriktionen:** Für die einzelnen Tester müssen die vorgegebene Stundenanzahl pro Tag, Mittagspausen sowie Wochenenden und Feiertage berücksichtigt werden.

### **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 zu unterstützen. Hierbei ist die Erstellung eines zulässigen Testplans, welcher die meisten der 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 die Nebenbedingungen (2)-(7) berücksichtigt und Schnittstellentests, welche gegen bestimmte Restriktionen verstoßen, einfach am Ende des Planungszeitraums eingeplant.

Das Problem einer derartigen Vorgehensweise ist allerdings, dass der so ermittelte Testplan i.d.R. sowohl gegen Restriktion (1) verstößt als auch die vorgegebenen Planungsziele außer Acht lässt. Bei einem derartigen Zeitplan existieren keine zusammenhängende Testzeiträume für die einzelnen Tester und auch der Gesamtzeitraum für die Durchführung der Tests ist inakzeptabel. In der Praxis einsetzbare und von den Testern akzeptierte Testpläne müssen alle Restriktionen erfüllen, als auch möglichst gut bezüglich der aufgeführten Zielfunktionen sein. Im Folgenden wird ein kurzer Überblick über mögliche Lösungsansätze gegeben.

### **3.1. Gegenwärtiger Lösungsansatz**

Beim Partnerunternehmen wurde die Aufstellung eines Zeitplans bisher durch einen menschlichen Planer mithilfe eines Tabellenkalkulationsprogramms (Excel) durchgeführt. Der Planungsprozess besteht aus der Ermittlung aller zu planenden Schnittstellen und der manuellen Einplanung aller Tests. Der menschliche Planer geht so vor, dass er zu Beginn des Planungsprozesses Komponenten mit vielen Schnittstellen so am Stück eingeplant, dass alle Restriktionen 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 eine Restriktion verletzt wird und gleichzeitig versucht, die Planungsziele aus Abschnitt 2.1 möglichst gut zu erreichen.

Das Hauptproblem bei der gegenwärtigen Planungsmethode liegt im 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 verstößt. Da gegenwärtig ca. 10 Testpläne pro Jahr erstellt werden, ist der Aufwand alleine für die Erstellung der entsprechenden Testpläne recht hoch.

### **3.2. Exakte Verfahren**

Es existiert eine Reihe von unterschiedlichen exakten Planungsverfahren, welche nach entsprechenden Anpassungen auch für das hier vorliegende Schnittstellenplanungsproblem eingesetzt werden können [Zäpf94,KäTe04]. Das Schnittstellenplanungsproblem kann hierbei als eine Variante des Maschinenbelegungsproblems formuliert werden. 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, Entschei-



dungsbaumsuchverfahren, begrenzte Enumeration, Cut-Verfahren oder Branch-and-Bound Ansätze verwendet werden. Das Hauptproblem bei der Verwendung von exakten Verfahren liegt in der Komplexität und Größe des vorliegenden Schnittstellenplanungsproblems. Da das Problem auf ein resource-constrained project scheduling Problem [Bruc99] zurückgeführt werden kann, können exakte Verfahren oft nur für kleine Probleminstanzen mit akzeptablem Aufwand eingesetzt werden.

### **3.3. Lösungsheuristiken**

Ein anderer Weg wird durch Lösungsheuristiken beschritten. Lösungsheuristiken (oft auch als Heuristiken bezeichnet) sind Verfahren, bei denen explizit auf die Sicherstellung der Ermittlung der optimalen Lösung verzichtet wird. Heuristiken werden oft in Anlehnung an praktisch handhabbare oder plausibel erscheinende Vorgehensweisen so konstruiert, dass sie Planungsprobleme schnell und mit vertretbarer Lösungsqualität lösen. Das Hauptproblem derartiger Verfahren liegt in der Ermittlung geeigneter heuristischer Vorgehensweisen, welche in akzeptabler 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.

### **3.4. Metaheuristiken**

Metaheuristiken (engl. „metaheuristics“) bauen auf Heuristiken auf und verwenden einfache, oft der Natur oder der Physik entlehnte Optimierungs- und Planungsprinzipien („naturinspiriert“). Oft ermöglichen sie eine zügige Modellierung und effiziente Lösung von Planungsproblemen.

Beispiele für Metaheuristiken sind lokale Suchverfahren (z. B. „Simulated Annealing“ oder „Tabu Search“), Evolutionäre Algorithmen [Gold89] oder auch kombinierte Verfahren (z. B. COSA [Wend95]). Derartige Verfahren wurden in den letzten Jahren mit Erfolg in der Produktionsfeinplanung [KuRo95, KSS95, ScMe00, WaZä00], 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 ist es notwendig, dass vollständige Lösungen des Planungsproblems (z. B. komplette Zeitpläne) so kodiert werden, dass geeignete Suchoperatoren angewandt und die vollständigen Lösungen bezüglich ihrer Qualität 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 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 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 der Ausgangslösungen.

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

## 4 Konzeption eines metaheuristikbasierten Planungssystems

Im Rahmen der Überlegungen zur Auswahl eines geeigneten Ansatzes für die Lösung des Schnittstellenplanungsproblems wurde ein Genetischer Algorithmus (GA) ausgewählt. 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 alle Nebenbedingungen ausreichend erfüllt wurden.

### 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. [Roth06]). 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, Roth06].

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  $k$  Testsystemen eingeplant werden. Hierbei wird jede einzuplanende Schnittstelle mit einer eindeutigen Nummer  $i \in \{0, 1, \dots, n-1\}$  und jedes Testsystem mit einer eindeutigen Nummer  $j \in \{0, 1, \dots, k-1\}$  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.

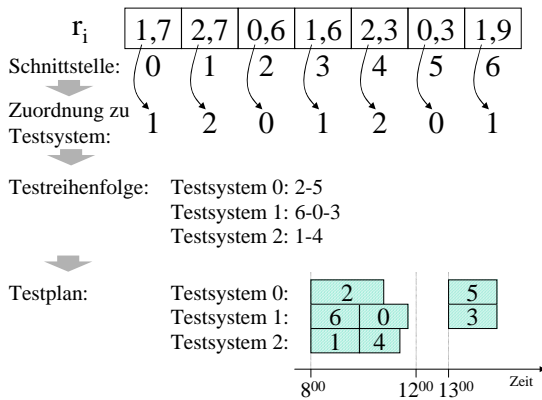


Abbildung 1: Erzeugung eines Zeitplans aus einer kodierten Lösung

Abbildung 1: Erzeugung eines Zeitplans aus einer kodierten Lösung folge 2-5 ( $r_2 > r_3$ ) 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 ( $r_1 > r_4$ ) 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  $k$  Testsystemen gewonnen werden. Durch eine derartige Kodierung wird ein Teil der Nebenbedingungen berücksichtigt. Da bei der Verwendung einer gewichteten Kodierung jede Schnittstelle nur ein einziges Mal eingeplant werden kann, wird jeder Schnittstellentest auch genau nur ein einziges Mal durchgeführt. Darüber hinaus kann auf einer Maschine zum gleichen Zeitpunkt auch nie mehr als ein Test durchgeführt werden, sodass automatisch Nebenbedingung (2) eingehalten wird.

Nachdem durch die Kodierung festgelegt worden ist, in welcher Reihenfolge die Tests auf den jeweiligen Testsystemen eingeplant werden, muss in einem zweiten Schritt aus der Reihenfolge ein Zeitplan erstellt werden. Bei dessen Erstellung werden die Restriktionen (7) wie z. B. maximale tägliche Arbeitszeit, Mittagspause oder Wochenenden berücksichtigt (vgl. Abbildung 1). Am Ende des Konstruktionsprozesses steht ein Zeitplan, welcher Nebenbedingungen (2) und (7) berücksichtigt.

## 4.2. Testplanbewertung

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 Reihen-

Beim Einsatz von Metaheuristiken muss die Qualität eines Testplans mithilfe einer Bewertungsfunktion bewertet werden. Aufgrund der vorhandenen unterschiedlichen Ziele liegt ein multikriterielles Problem vor [Ehr05], bei dem paretooptimale Lösungen bezüglich der vorgegebenen Planungsziele ermittelt werden können. Prinzipiell sind populationsbasierte Metaheuristiken gut für die Lösung von 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

Abbildung 2: Gewichtungen für Zeitplanbewertung und Parameter für GA

gen additiv zusammengefasst werden.

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 erzeugten Zeitpläne. Je größer

die jeweilige Gewichtung gewählt wird, desto stärker wird dieses Ziel bzw. die Einhaltung einer Restriktion berücksichtigt. Die Bewertung eines Zeitplans hängt von folgenden Kriterien ab:

**Zusammenhängende Testzeiträume/Unterbrechungsfreies Testen:** Die Bewertung eines Zeitplans wird um  $B_{\text{testzeitraum}} = U_{\text{testzeitraum}} (l_{\text{testzeitraum}} / n)$  erhöht, wobei  $l_{\text{testzeitraum}}$  die kumulierte Anzahl der Schnittstellen bezeichnet, welche jeweils ohne Unterbrechung durch einen Tester getestet werden können, und  $n$  die Gesamtanzahl der einzuplanenden Schnittstellen angibt.

$U_{\text{testzeitraum}}$  kann durch den Entscheider vorgegeben werden (vgl. Abbildung 2).

**Frühzeitiger Test von Basiskomponenten:** Die Bewertung eines Zeitplans wird um die Teilbewertung  $B_{\text{basis}} = U_{\text{basis}} (1/l_{\text{basis}} \sum_{\text{Basiskomp.}} (1 - (pos_i / n)))$  erhöht, wobei  $l_{\text{basis}}$  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  $B_{testtage} = U_{testtage}^{fix}$  erhöht. Ansonsten gilt  $B_{testtage} = U_{testtage}^{teil} (2 - (t - t_{min}))$ , 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 auftreten, wird die Bewertung eines Zeitplans um  $B_{Überschneidung} = U_{Überschneidung}^{fix}$  erhöht. Ansonsten gilt  $B_{Überschneidung} = U_{Überschneidung}^{teil} (1 - (n_{ii} / n))$ , wobei  $n_{ii}$  die Gesamtanzahl der Überschneidungen bezeichnet.

**Zeitrestriktionen für Tester:** Falls keine Zeitrestriktionen verletzt werden, gilt  $B_{Zeit} = U_{Zeit}^{fix}$ . Ansonsten ergibt sich  $B_{Zeit} = U_{Zeit}^{teil} (1 - (n_z / m_z))$ , 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 entsprechender  $U$  werden die einzelnen Ziele und Restriktionen unterschiedlich stark gewichtet und dementsprechend mögliche Lösungen (Zeitpläne) unterschiedlich gut bewertet.

### 4.3. Initialisierung von Startlösungen

Bei der Verwendung von Metaheuristiken 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 sowohl vorhandenes Vorwissen über Eigenschaften von qualitativ guten Lösungen als auch bestehende Nebenbedingungen berücksichtigt werden. Von beiden Möglichkeiten wurde bei der Lösung des Schnittstellenplanungsproblems 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 werden bei der Initialisierung auch schon die Nebenbedingungen (5) und (6) berücksichtigt. 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 (5) oder (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 „Leerschnittstel-

len" 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 die 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 GA [Gold89] gewählt. GA wenden Suchoperatoren (Rekombination und Mutation) auf eine Menge (Population) von Problemlösungen (Individuen) über mehrere Iterationen (Generationen) an. Die Verwendung von Rekombinations- und Mutationsoperatoren führt hierbei zu einer höheren Diversität der Lösungen, wohin gehend Selektion zu einer Intensivierung der Suche führt. GA wurden schon mit Erfolg für ähnliche Planungsprobleme eingesetzt [KSS95, KuRo95, WaZä00] und sind einfach anwend-

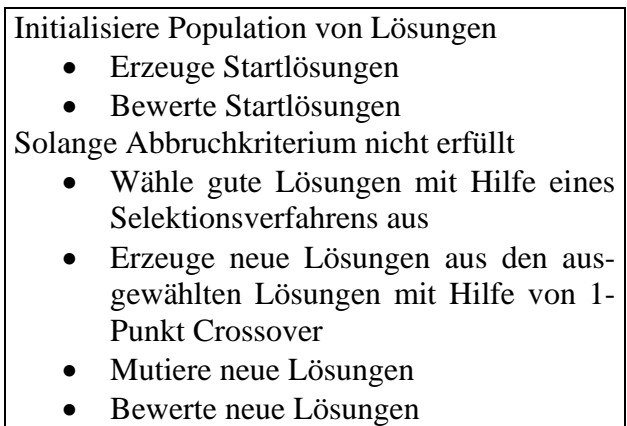


Abbildung 3: Ablaufschema eines GA

bar. Das Ablaufschema des verwendeten GA ist in Abbildung 3 dargestellt. Als Abbruchkriterium wurde eine maximale Anzahl an Suchschritten bzw. eine maximal zur Verfügung stehende Rechenzeit gewählt (vgl. hierzu Abbildung 2).

Als Rekombinationsverfahren wurde 1-Punkt Crossover mit Wahrscheinlichkeit  $P_{cross}$  gewählt. 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 sichergestellt, dass Restriktionen (5) und (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 gewählt (üblicherweise  $\mu = 200$  und  $\lambda = 800$ ). Dies bedeutet, dass aus  $\mu$  Ausgangslösungen jeweils  $\lambda$  neue Lösungen erzeugt werden. Anschließend werden aus den  $\mu + \lambda$  Lösungen dann wieder die  $\mu$  besten Lösungen ausgewählt.



## 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.

Neben dem eigentlichen Planungssystem zur Ermittlung geeigneter Zeitpläne 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 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.

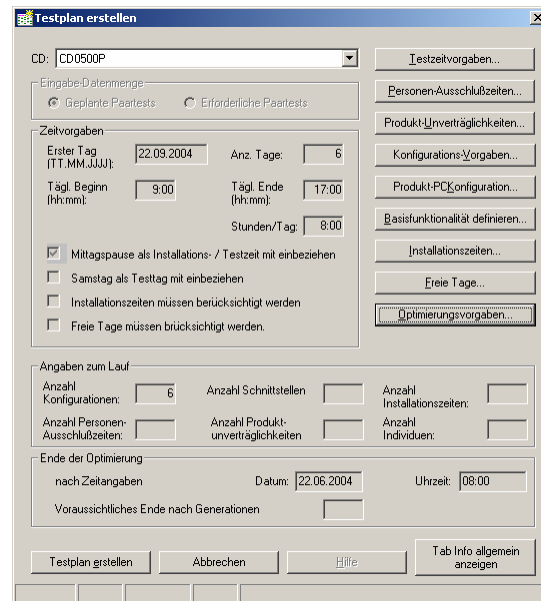


Abbildung 4: Hauptdialog Planungswerkzeug

## 6 Zusammenfassung

Bei der Erstellung von komponentenbasierten Informationssystemen muss im Rahmen der Qualitätssicherung nicht nur die Funktionalität der einzelnen Komponenten sondern auch das korrekte Zusammenspiel unterschiedlicher Komponenten überprüft werden. Hierzu müssen Schnittstellentests an zentraler Stelle durchgeführt und ein entsprechender Zeitplan für die Durchführung dieser Tests 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 Kodierung und Erzeugung von Zeitplänen, die Bewertung der Qualität von Testplänen und die Erzeugung von Startlösungen. Als Metaheuristik für die automatisierte Zeitplanerzeugung wird ein GA 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.

## Literaturverzeichnis

- [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.
- [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.
- [Bruc99] Brucker, P.; Drexl, A.; Möhring, R.; Neumann, K.; Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods, European Journal of Operational Research 112, 1999, S. 3-41.
- [Brü95] Brüggemann, W.: Ausgewählte Probleme der Produktionsplanung: Modellierung, Komplexität und neuere Lösungsmöglichkeiten. Physica, Heidelberg, 1995.
- [Deb01] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester, UK, 2001.
- [Ehrg05] Ehrgott, M.: Multicriteria Optimization. Springer, Berlin, 2005.
- [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.
- [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.
- [Roth06] Rothlauf, F.: Representations for genetic and evolutionary algorithms. 2. Auflage, Heidelberg, Springer, 2006.
- [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 manufacturing, *International Journal of Production Economics* 68, 2000, S. 199-213.
- [Wend95] Wendt, O.: Tourenplanung durch Einsatz naturanaloger Verfahren. Promotion. Wiesbaden, Gabler, 1995.
- [Zäpf94] Zäpfel, G.: Entwicklungsstand und –tendenzen in PPS-Systemen. In: *Handbuch Produktionsmanagement*. Hrsg.: Corsten, H., Gabler, Wiesbaden, 1994, S. 719-745.