Qualitätssicherung lagerlogistischer Steuerungssoftware durch Simulation

Kai Gutenschwager, Karl-August Fauth, Sven Spieckermann, Stefan Voß

Summary on page 46

In der Lagerlogistik wird die diskrete, ereignisorientierte Simulation im wesentlichen zur Layoutplanung und Definition der Steuerungsstrategien komplexer Läger eingesetzt. In diesem Artikel stellen wir Möglichkeiten der weitergehenden Nutzung von Simulationsmodellen zur Unterstützung von Tests der Steuerungssoftware großer Lagersysteme am Beispiel des Warenumschlaglagers Cargo-Hub-2001 der Cargologic, Zürich, vor. Diese basieren auf einer Kopplung der beiden Systeme, aufgrund derer sich Back-to-back-Tests auf Modulebene durchführen lassen. Ziel einer solchen Kopplung ist eine Verkürzung der kostenintensiven Inbetriebnahmezeit des Gesamtlagersystems.

Simulation, Softwaretests, Software-Qualitätssicherung, Leitrechner, Lagersysteme, Logistik

1 Einleitung

Diskrete, ereignisorientierte Simulation wird seit über zwei Jahrzehnten zur Unterstützung der Planung logistischer Systeme eingesetzt [11,12,18]. Die sich oft gleichende Aufgabe besteht in der Überprüfung des Systemdurchsatzes, d. h. ob die gewünschte Ausbringungsmenge pro Zeiteinheit abgegeben werden kann. Diese Frage wird im Rahmen der Analyse so unterschiedlicher Systeme wie Automobilrohbau oder Distributionszentren untersucht; vgl. z. B. [7,26]. Während der Konzeption eines logistischen Systems werden die weitgehend statischen Berechnungen und Zeichnungen des Planungsingenieurs durch Simulationsexperimente unterstützt, die ein Testen (und Nachweisen) der Leistungsfähigkeit dynamischer Systeme ermöglichen sowie die dynamischen Zusammenhänge anschaulich am Bildschirm aufzeigen.

Seit einigen Jahren gibt es, nicht zuletzt dank der zunehmenden Möglichkeiten der zahlreichen Simulationswerkzeuge [23, 21], zusätzlich einen Trend zu sog. „Online-Simulation“ [22]. So propagiert die den Einsatz der Simulation in Deutschland beschreibende Richtlinie 3633 des Vereins Deutscher Ingenieure [24] den Einsatz von Simulation während des gesamten Lebenszyklus eines logistischen Systems: Neben der Systemplanung wird der Einsatz von Simulation auch während der Realisierung und Inbetriebnahme und zur Unterstützung des laufenden Betriebs des Systems empfohlen.

In dieser Arbeit befassen wir uns mit dem Einsatz von Simulationsmodellen während der Realisierung und Inbetriebnahme großer Lagersysteme. Ziel des beschriebenen Ansatzes ist es, die kostenintensive Einführungsphase der Software durch die durchgängige Nutzung des Simulationsmodells während der Testphase drastisch zu verkürzen. Die Kosten entstehen, da das Aufdecken und Beheben von Fehlern in der Software durch „Experimente“ an der realen Anlage sehr aufwendig ist.

In Abschn. 2 wird zunächst die Entwicklung von Steuerungssoftware für Lagersysteme unter besonderer Berücksichtigung von Testverfahren vorgestellt, um darauf aufbauend Potentiale und Voraussetzungen für den Einsatz der Simulation im Rahmen

Kai Gutenschwager[1,3], Karl-August Fauth[2], Sven Spieckermann[3], Stefan Voß[1]
[1]Technische Universität Braunschweig, Institut für Wirtschaftswissenschaften, Abt-Jerusalem-Straße 7, D-38106 Braunschweig
[2]ALSTOM (ehemals Cegelec AEG) Anlagen- und Automatisierungstechnik GmbH, Lyoner Straße 9, D-60528 Frankfurt
[3]SimPlan Gesellschaft für Simulation betrieblicher Abläufe mbH, Edmund-Seng-Straße 3–5, D-63477 Maintal-Dörnigheim

von Software-Tests in Abschn. 3 zu diskutieren. Als Beispiel der vorgestellten, verallgemeinerten Vorgehensweise für simulationsgestützte Softwaretests dient ein Projekt der SimPlan GmbH und der ALSTOM AT GmbH (Abschn. 4). Eine Bewertung des vorgestellten Konzepts anhand dieses Beispiels findet sich in Abschn. 5.

2 Entwicklung von Steuerungssystemen in der Lagerlogistik

Steuerungssysteme für große Lagersysteme sind meist sehr komplex. Die Steuerung der Abläufe in einem Lager wird daher generell, wie in der Automatisierung großer Anlagen üblich, in mehrere, hierarchisch strukturierte Ebenen unterteilt [9,25]. Die sog. dispositiven Entscheidungen, z. B. die Wahl von Lagerfächern, die Priorisierung von Ein- und Auslagerungsaufträgen oder die Auftragswahl für Fahrzeuge, werden von überlagernden Materialfluß- oder Lagerverwaltungsrechnern getroffen. Die den physischen Materialfluß konkret realisierenden ausführenden Entscheidungen, z. B. das Ansteuern von Antrieben oder Verzweigungsentscheidungen anhand bereits erfolgter Zielvorgaben, werden dann (auf einer unteren Ebene) von speicherprogrammierbaren Steuerungen (SPS) übernommen, die lange Zeit lediglich in einer Assembler-nahen Sprache programmiert werden konnten. Moderne SPS lassen sich auch in einer Hochsprache programmieren. Für die Steuerung mobiler Anlagekomponenten kommen spezielle Industrie-PC-Systeme zum Einsatz.

Die Realisierung der Steuerungssoftware von Lagersystemen stellt aufgrund der Komplexität der Systeme in der Praxis meist ein umfangreiches Software-Engineering-Projekt dar [19]. Der Entwurf dieser Software-Systeme wird idealtypisch im Anschluß an die Konzeption des Lagers durchgeführt, ausgehend von einem Layout des Lagers und der Definition der einzusetzenden Steuerungsstrategien. Diese werden in der Analyse- und Definitionsphase für das Softwaresystem konzipiert (Produktdefinition, [20]).

Die Anforderungen, die in der Produktdefinition des Steuerungssystems festgelegt sind, unterscheiden sich in Lagersystemprojekten von reinen Softwareprojekten, da sich die Anforderungen des Auftraggebers in der Regel nicht allein auf die Funktionalität der Software, sondern vielmehr auf die Funktionalität des gesamten Lagersystems beziehen. Der Abnahmetest, der vor dem tatsächlichen Einsatz steht, bezieht sich nicht nur auf die Software an sich, sondern auch auf betriebswirtschaftliche oder technische Kenngrößen, wie z. B. den Systemdurchsatz. Die Software kann daher formal fehlerfrei sein, der Abnahmetest aber dennoch negativ ausfallen. Es kann erst in der Testphase der Software festgestellt werden, ob die Produktdefinition mit (logischen) Fehlern behaftet ist, sofern die Funktionalität des Gesamtsystems nicht zu Beginn des Softwareprojekts überprüft wird. Solche Fehler in einer späten Phase zu entdecken, kann die Kosten für frühe Tests bis zum Faktor 100 übersteigen [2]. Hier kommen Simulationsstudien ins Spiel, mit denen eine Aussage über das Gesamtsystem am Modell erarbeitet werden kann, wobei neben der physischen Anlage auch die notwendigen Steuerungsfunktionalitäten abgebildet werden, um verschiedene Szenarien (z. B. Auftragslagen) zu überprüfen.

In reinen Softwareprojekten entspricht diese Vorgehensweise weitestgehend der Methode des (Wegwerf-)Prototyping zur Überprüfung der Produktdefinition relevanter Systemteile [17]. Die meisten Simulationsmodelle weisen einen solchen Wegwerfcharakter auf: Sobald die Analyse- und Definitionsphase abgeschlossen ist, wird das Simulationsmodell nicht mehr eingesetzt. Die Modelldokumentation einer solchen Studie wird in vielen Praxisprojekten bestenfalls zum Abgleich mit der Produktspezifikation der Steuerungssoftware eingesetzt, z. B. im Rahmen von Audits oder Reviews. Alle weiteren Testaktivitäten werden meist unabhängig von der Simulationsstudie durchgeführt. Diese betreffen im wesentlichen die Modultests, die Integrationstests und die Systemtests.

Testverfahren werden in Black-box- und White-box-Verfahren unterschieden. Bei White-box-Verfahren, mit denen wir uns im weiteren nicht beschäftigen werden, wird die interne Struktur des Programms, der Programmcode, auf Fehler untersucht.

Ansatzpunkt der Black-box-Verfahren ist nicht die Struktur des Programms, sondern dessen Spezifikation, d. h. es werden lediglich Funktionen dahingehend überprüft, ob die gewünschten Ausgaben zu bestimmten Eingabedaten erfolgen. Da in der Regel nicht alle Fälle überprüft werden können, kann man ein Programm nicht so testen, daß seine

Fehlerfreiheit garantiert werden kann. Um eine geeignete Auswahl an Testfällen zu erhalten, bietet es sich an, Testfälle zu Klassen zusammenzufassen (funktionale Äquivalenzklassenbildung) und nur Vertreter aller Klassen zu überprüfen oder reine Zufallstests durchzuführen. Eine weitergehende Klassifikation und Übersicht zu diesen Verfahren findet sich in [13].

In allen Testphasen wird empfohlen, Kombinationen von White-box- und Black-box-Verfahren anzuwenden. Dies betrifft natürlich auch die Entwicklung und Inbetriebnahme von Steuerungssoftware in der Lagerlogistik. Hier werden für die unterlagerten SPS z. T. eigenständige Testumgebungen eingesetzt, die diese Module durch einfache Simulationssysteme testen [10].

Auch Modultests lassen sich prinzipiell unabhängig voneinander durchführen. Um hier Black-box-Verfahren anwenden zu können, ist es notwendig, Testfälle für die einzelnen Module zu generieren. Rufen diese Module wiederum andere Module als Subroutinen auf, so sind diese entweder zunächst zu implementieren und zu testen (Bottom-up-Ansatz) oder geeignete Rückgabewerte (oder Aktionen) dieser durch Implementierung der Module ohne deren vollständige Funktionalität, sog. *Stub*-Module (englisch für Stumpf, Stummel), bereitzustellen. Besteht eine eindeutige Hierarchie der Module (bzgl. der Aufrufe), so kann auf die Implementierung von Stub-Modulen beim Bottom-up-Ansatz vollständig verzichtet werden, beim Top Down-Entwurf würde hingegen für jedes Modul mit Ausnahme der Module auf der höchsten Hierarchieebene ein äquivalentes Stub-Modul implementiert werden müssen. Testfälle müssen dann aber nur für die Module der höchsten Ebene generiert werden, so daß immer das Gesamtsystem überprüft wird.

In beiden Ansätzen wird versucht, Module inkrementell zu testen. Wurde beim Bottom-up-Ansatz ein Modul getestet, so wird dieses auch bei den Tests der aufrufenden Module weiterhin eingesetzt und somit getestet. Dies entspricht einem schrittweisen Integrationstest. Weitere Vor- und Nachteile der beiden Ansätze werden in [15] ausführlich dargestellt. Im Gegensatz dazu ist es auch denkbar, zunächst alle Module einzeln zu testen und erst abschließend einen Integrationstest durchzuführen. Myers [15] spricht in diesem Fall von einem „Big-bang-Test“. Diese Alternative ist aufgrund der Komplexität des betrachteten Testobjekts und des sich ergebenden Problems der Fehlerlokalisierung aber denkbar ungünstig.

Ein wesentliches Problem des inkrementellen Testens ist neben der Programmierung der Stub-Module die Bestimmung der nach der Spezifikation erwarteten Ausgabedaten zu den (ggf. zufällig) erzeugten Testeingabedaten. Einen Ausweg bieten sog. Back-to-back-Tests an [14]. Hier werden verschiedene Versionen eines Programms betrachtet, die aus derselben Spezifikation gewonnen werden und somit eine identische Funktionalität besitzen. Zwei Programmversionen erzeugen für einen Testfall dann entweder gleiche Ausgaben (d. h. beide sind für diesen Testfall korrekt, oder beide sind nicht korrekt) oder unterschiedliche Ausgaben (d. h. eines ist korrekt, das andere ist inkorrekt oder beide sind inkorrekt). Bei unterschiedlichen Ausgaben ist eine Fehlersuche unumgänglich.

Die Entwicklung mehrerer Programmversionen führt zu erheblichen Entwicklungskosten. Bei n unabhängig voneinander entwickelten Versionen erhöht sich der Aufwand um den Faktor n. Zusätzlich fallen Kosten zur Einbettung der Versionen in eine gemeinsame Testumgebung an. Der Vorteil dieser Vorgehensweise liegt darin, daß eine große Anzahl vergleichender Tests automatisch durchgeführt werden kann (sog. Massentests, [13]). Diese Vorgehensweise kann damit auch aus ökonomischer Sicht durchaus interessant sein.

Die Testumgebung sollte es erlauben, einzelne Module mit den verschiedenen Programmversionen zu koppeln, um Unterschiede in den Testergebnissen direkt auf diese zurückführen zu können. Ist dies nicht möglich, so ist eine Fehlerbehebung weit komplizierter. Lassen sich einzelne Module in die Umgebung des Zweitsystems integrieren, so können Module inkrementell getestet werden, wobei keine Stub-Module mehr programmiert werden müssen und dennoch immer an einem Gesamtsystem getestet werden kann. Die Vorteile des Bottom-up- und des Top-down-Ansatzes lassen sich so verbinden.

In Lagerlogistikprojekten wird das Zusammenspiel der einzelnen Systemkomponenten (Integrationstests) heute in der Regel erst an der realen Anlage überprüft. Die Inbetriebnahme der Software stellt aus Sicht der Softwareentwicklung aber einen wesentlichen Teil der Testphase dar. Brooks [5] gibt ei-

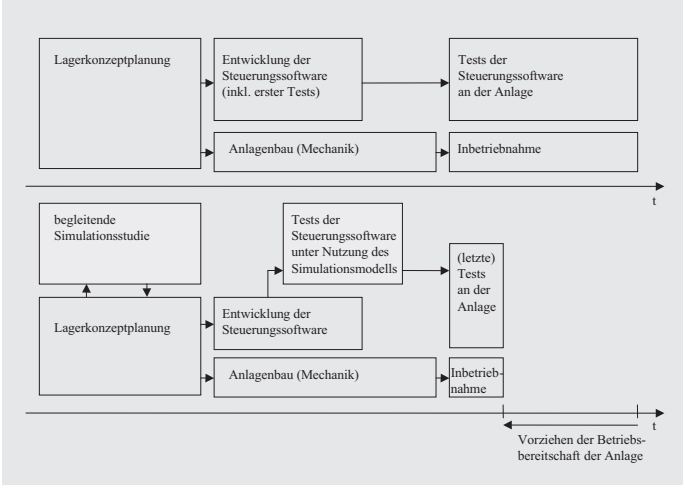

Abb. 1. Verkürzung der Gesamtprojektdauer durch simulationsgestützte Softwaretests

nen Erfahrungswert von 25 %, Myers [15] sogar von 50 % für den zeitlichen Anteil der Testphase am Gesamtprojekt an. Im folgenden Abschnitt sollen daher Potentiale und Voraussetzungen für den Einsatz der Simulation in der Testphase diskutiert werden.

3 Kopplung von Simulationsmodell und Steuerungssystem zur durchgängigen Software-Qualitätssicherung

3.1 Unterstützungspotentiale der Simulation

Kann mit wesentlichen Teilen der Testphase der Steuerungssoftware erst begonnen werden, sobald das reale Lager aufgebaut ist, so ist bei einer hohen Projektdauer davon auszugehen, daß eine weitreichende und damit kostenintensive Zeitspanne zwischen Fertigstellung der physikalischen Anlage und der Nutzung entsteht. Allein aus diesem Grund erscheint es sinnvoll, Simulationsmodelle auch während der Testphase, die dann parallel zur Erstellung der physikalischen Anlage erfolgen kann, einzusetzen (vgl. Abb. 1).

Abb. 1 ist zu entnehmen, daß nach Abschluß der Lagerkonzeptplanung bei Nutzung eines Simulationsmodells schon während der Entwicklung der Steuerungssoftware mit der Testphase begonnen werden kann. Eine Unterbrechung der Testphase tritt hier ggf. auf, da letzte Tests immer am Lagersystem selbst durchzuführen sind. Liegt kein Simulationsmodell vor, so wird ein wesentlicher Teil der Testphase erst begonnen, wenn die Anlage fertiggestellt ist. Durch den Einsatz der Simulation läßt sich die Gesamtprojektdauer neben dem Vorziehen der Testaktivitäten auch dadurch verkürzen, daß sich die Experimentdurchführung am Computer als deutlich einfacher und schneller erweist.

Das allgemeine V-Modell[1] von Boehm [1] bietet einen geeigneten Ausgangspunkt zur Darstellung der Unterstützungsfunktion der Simulation in Softwareprojekten (vgl. Abb. 2).

Abb. 2 soll verdeutlichen, daß den einzelnen konstruktiven Aktivitäten der Softwareerstellung (Produktdefinition, Spezifikation und Implementierung) die prüfenden Aktivitäten der Qualitätssicherung gegenübergestellt werden können. Testfälle für die entsprechenden Testphasen bauen dabei aufeinander auf, wodurch der dargestellte Fluss von Testfällen entsteht.

Die Phasen der Produktdefinitions- und Softwarespezifikationstests können durch eine die Gesamtplanung begleitende Simulationsstudie unterstützt werden, wobei Testfälle für Abnahmetests bereits in der ersten Projektphase festgelegt und überprüft werden können. Die Modelldokumentation kann z. B. im Rahmen von Reviews für die ersten Testaktivitäten herangezogen werden.

Mit dem Simulationsmodell liegt darüber hinaus bereits eine erste lauffähige Programmversion

[1] Ausgehend von diesem einfachen, für unsere Betrachtungen aber ausreichendem Modell wurde ein V-Modell zur Standardisierung der Software-Entwicklung für die Bundeswehr und Behörden [4] und hierauf aufbauend das V-Modell 97 [6] zur Planung und Durchführung von IT-Projekten entwickelt, das einen Rahmen zur (objektorientierten) Software-Entwicklung vorgibt.

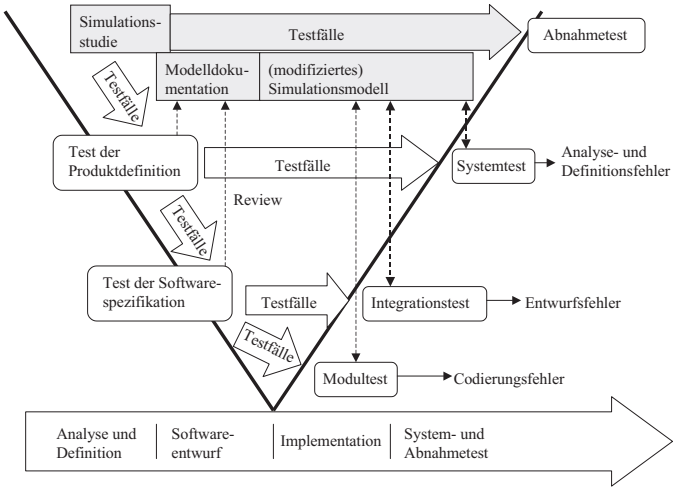

Abb. 2. Einordnung der Simulation in das V-Modell. (Nach Boehm [1])

des Steuerungssystems vor, da sich das Simulationsmodell in ein Modell der Anlage und den Steuerungsteil, welcher der zu entwickelnden Steuerungssoftware entspricht, aufteilen läßt. Diese Version kann unter bestimmten Voraussetzungen (Modellmodifikationen) für Back-to-back-Tests mit der Steuerungssoftware eingesetzt werden, wobei die angesprochenen hohen Kosten der Entwicklung des Zweitsystems natürlich weitestgehend entfallen, wenn das Simulationsmodell bereits in der Planungsphase des Gesamtlagers erstellt wurde (vgl. Abb. 2).

Wurde das Simulationsmodell selbst eingehend getestet, so können mittels dieses Modells erwartete Werte oder Resultate auf einfache Weise generiert und die Korrektheit der Testfälle somit quasi sichergestellt werden. Zugleich sind diese Testfälle reproduzierbar, da das identische Simulationsexperiment mit korrigierten Programmtexten wiederholt werden kann. In der Praxis finden sich trotz dieser Vorteile nur wenige Anwendungen einer solchen Kopplung [3,16].

Um ein Simulationsmodell, das zur Planungsunterstützung erstellt wurde, auch für weitergehende Back-to-back-Tests zu nutzen, bietet sich die angesprochene Online-Kopplung der beiden Systeme an. Noche [16] beschreibt zwei Einsatzfälle bei der erweiterten Nutzung der Simulationsumgebung: Tests auf Leitrechnerebene (Integrationstests) und Funktionstests auf SPS-Ebene. Die Unterstützung der Integrationstests bezieht sich dabei auf das Zusammenspiel der miteinander kommunizierenden Rechner, also auf den Telegrammverkehr (korrekte Adressierung, richtige Antworttelegramme etc.). Der Integrationstest entspricht hierbei einer Big-bang-Vorgehensweise: Das Testobjekt stellt die vollständige Steuerungssoftware dar, die alle dispositiven Entscheidungen trifft, während das Simulationsmodell lediglich die Anlage und Fahrbewegungen mit den notwendigen Telegrammen an die Steuerungssoftware abbildet.

Unter der Voraussetzung, daß sich einzelne Module der Steuerungssoftware in das Simulationsmodell integrieren lassen, können durch eine Kopplung darüber hinaus auch Modultests durchgeführt werden. Im Verlauf eines Simulationsexperiments werden dabei automatisch Testfälle für das zu untersuchende Testmodul generiert. Eine Bewertung der Funktionalität eines Moduls kann durch den direkten Vergleich mit dem Modul des Zweitsystems (hier als Teil des Simulationsmodells) erfolgen, bzw. auch durch einen weiterhin fehlerfreien Verlauf des Simulationsexperiments.

Auf diese Weise können Module inkrementell getestet werden, wobei keine Stub-Module programmiert werden müssen. Der wesentliche Vorteil liegt darin, daß immer an einem Gesamtsystem getestet werden kann und die Anforderungen des Abnahmetests in jedem Schritt der Modultests überprüft werden können. Gerade bei kleinen Unterschieden zwischen Simulationsmodell und Steuerungssoftware oder Modifikationen der Spezifikation kann diese Vorgehensweise von großer Bedeutung bei der Suche nach „Fehlern“ sein, die zu einer verminderten Leistung des Gesamtsystems führen.

Ein weiterer Vorteil ist, daß Massentests durchgeführt werden können. Allerdings bedeutet die (möglicherweise zu starke) Einschränkung bezüglich der Breite der zufällig erzeugten Testfälle, daß

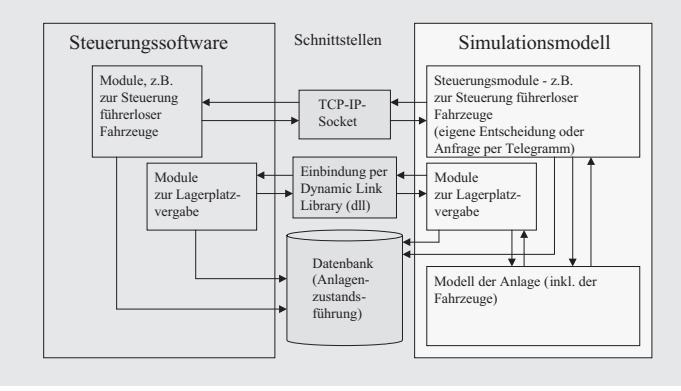

Abb. 3. Kopplung von Steuerungssoftware und Simulationsmodell

die Simulation nur unterstützend eingesetzt werden kann, nicht aber alle anderen Tests (v. a. White-box-Verfahren) zu ersetzen vermag. Zumindest lassen sich die Testfälle ex post aber Klassen zuordnen, sobald sie modulbezogen ausgewertet werden.

So können bei einer Online-Kopplung z. B. die Steuerungskomponente führerloser Fahrzeuge und die Lagerplatzvergabe (für einzulagernde Paletten) zunächst einzeln getestet werden, indem diese über eine Schnittstelle vom Simulationsmodell bei Bedarf aufgerufen werden. Der nächste Schritt besteht dann darin, beide parallel zu testen und darüber hinaus auch zu prüfen, ob z. B. der Anlagenzustand in der Datenbank korrekt mitgeführt wird. In der letzten Stufe der Integrationstests dient das Simulationsmodell idealtypisch nur noch dazu, die Physik der Anlage und die Testfälle, z. B. ganze Ein- und Auslagerungsaufträge, abzubilden.

Das Simulationsmodell bietet auch für Systemtests eine ideale Basis, in denen v. a. Laufzeit- und Stresstests durchgeführt werden, welche die Effizienzanforderungen validieren sollen, da Testfälle hier in der Regel wesentlich schneller als in Realzeit überprüft werden können. Noche [16] gibt allerdings an, daß für telegrammbasierte Tests eine Realzeitsimulation notwendig ist. Die Möglichkeit, ein Experiment zu unterbrechen, bis das Ergebnis einer Anfrage (das Antworttelegramm) vorliegt, kann aber dazu genutzt werden, weitaus schneller als Realzeit zu simulieren, da auf diese Weise keine Synchronisationsprobleme bezüglich des Anlagenzustands entstehen. Sind aber gerade diese Synchronisationsprobleme Untersuchungsgegenstand der Tests, so ist ein Realzeitbetrieb des Simulationsexperiments erforderlich.

3.2 Voraussetzungen für die Kopplung

Um eine Kopplung zu realisieren, ist es zunächst notwendig, daß beide Systeme miteinander kommunizieren können. Zum einen müssen Funktionsaufrufe über die Systemgrenzen hinweg erfolgen können, zum anderen ist zu gewährleisten, daß bei beiden Systemen ein einheitlicher Anlagenzustand als Basis für alle dispositiven Entscheidungen vorliegt, wenn die Kopplung bereits inkrementellen Modultests dienen soll. Zur Kopplung sind mehrere Alternativen denkbar (vgl. Abb. 3):

- Von der Simulationsumgebung werden Funktionen (Module) der Steuerungssoftware über eine Schnittstelle zur Laufzeit eingebunden („dynamic link libraries"). Hier ist es notwendig, daß sich die Architektur des Simulationsmodells und der Steuerungssoftware weitestgehend entsprechen.
- Wird in der Steuerungssoftware ein Telegrammverkehr zwischen den einzelnen Elementen (Modulen und Anlagenteilen) eingesetzt, so kann das Simulationsmodell in diese Architektur integriert werden. Basiert die Kommunikation z. B. auf dem Kommunikationsprotokoll TCP/IP, so stellt die Simulationsumgebung einen sog. Socket bereit, der eine einfache Einbindung (Versenden und Empfangen von Telegrammen) ermöglicht.
- Greifen die zu testenden Module auf eine Datenbank mit dem Anlagenzustand zu, so ist es prinzipiell notwendig, daß alle Zustandsänderungen, die sich im Simulationsmodell ergeben, auch in die Datenbank übernommen werden, und umgekehrt. Einige Simulationsumgebungen bieten hierfür etwa eine ODBC-Schnittstelle (eine standardisierte SQL-Schnittstelle) an. Zur Synchronisation ist es entweder notwendig, daß auch das Simulationsmodell nur auf die Datenbank zugreift, wobei die bisherige interne Anlagenzustandsführung durch entsprechende Datenbankzugriffe

ersetzt werden muß oder diese Änderungen dem Simulationsmodell mitgeteilt werden (und dieses die Änderungen in der eigenen Anlagenzustandsführung nachvollzieht).

Zur Kopplung ist ein Simulationssystem erforderlich, das alle notwendigen Kommunikationsstrukturen abbilden kann. Eine Kopplung ist ferner erst realisierbar, wenn das Simulationsmodell ähnlich aufgebaut ist wie das reale System. Diese Ähnlichkeit betrifft sowohl den Detaillierungsgrad des Modells als auch die Systemarchitektur, speziell die Kommunikationsstruktur.

Ein Beispiel soll diese Forderung veranschaulichen: In einem Simulationsmodell seien die Gassen eines Lagers als Black box abgebildet, d. h. die Ein- und Auslagerungsvorgänge per Regalbediengerät (RBG) werden nur durch Zeiten abgebildet. Tatsächliche Wege und Lagerfächer sind im Modell nicht hinterlegt. Um jetzt die Steuerung auf ihre Leistung (z. B. Durchsatz pro Stunde) hin zu testen, ist aber eine Abbildung der Wege, Lagerfächer und Geschwindigkeiten des RBG notwendig, um die Fahraufträge im Simulationsmodell überhaupt verstehen (und nutzen) zu können.

Der notwendige Detaillierungsgrad wird somit durch die vollständige Abbildung aller Objekte, die in der Realität (per Schnittstelle) angesprochen werden, definiert. Es ist also sinnvoll, bereits in der Planungssimulation die später einmal zu testenden Module zu definieren.

Die Systemarchitektur sollte darüber hinaus ähnlich sein, so daß es äquivalente Module im Steuerungssystem und im Simulationsmodell gibt. Für eine Integration des zu testenden Moduls in das Simulationsmodell sind zunächst die sich ergebenden Schnittstellen zu berücksichtigen, d. h. Funktions- oder Telegrammaufrufe müssen dieselben Parameter und Rückgabewerte beinhalten. Im Modell werden Module durch Funktionsaufrufe miteinander kommunizieren, in der Steuerungssoftware ggf. durch einen Telegrammverkehr.

Ist bereits zu Beginn einer Simulationsstudie bekannt, daß eine Kopplung erfolgen soll, so können die genannten Punkte beim Entwurf des Simulationsmodells berücksichtigt werden. Sind die genannten Voraussetzungen erfüllt, so kann mit der Kopplung und dem Test begonnen werden. Auf die erfolgreiche Durchführung eines solchen Projekts soll im nächsten Abschnitt eingegangen werden.

4 Beispiel: Test der Steuerungssoftware des Warenumschlaglagers Cargo-Hub-2001

4.1 Systembeschreibung

Bei dem Warenumschlaglager der Cargologic am Flughafen Zürich handelt es sich um ein komplexes, relativ neuartiges Lagersystem (eine Übersicht zu Lagersystemen findet sich bei [8]). Die Anforderungen von Seiten des Betreibers bezogen sich auf den Durchsatz (Anzahl Einlagerungen und Auslagerungen an Paletten pro Stunde) und das Reaktionszeitverhalten, welches als die Differenz zwischen Anforderungszeitpunkt einer Palette und tatsächlicher Auslagerung definiert ist. Um diesbezügliche Leistungszusagen geben zu können, wurde eine Simulationsstudie durchgeführt [7]. Die Simulationsexperimente dienten vorrangig der Definition und iterativen Verbesserung der Steuerungsstrategien. Die Studie umfaßte auch den Entwurf von Testfällen (für den Abnahmetest). Das Simulationsmodell wurde mit SiMPLE ++ entwickelt, einem auf dem Bausteinkonzept basierenden, objektorientierten Simulationssystem. SiMPLE ++ ist eine Software der Tecnomatix, ehemals AESOP, Stuttgart. Für einen Systemvergleich wird auf Swain [23] und Spieckermann et al. [21] verwiesen.

Das Layout des Palettenlagers ist in Abb. 4 schematisch dargestellt. Ein- und Auslagerungen spielen sich (vereinfacht dargestellt) wie folgt ab: Für einzulagernde Paletten wird, nachdem sie über Bodenfördertechnik zu den Bereitstellplätzen transportiert worden sind, zunächst ein Lagerfach ermittelt und reserviert. Der weitere phyische Transport der Palette in das Lager erfolgt dann zunächst per Fördertechnik (z. B. Rollenbahnen) zu einem Lift, der die Palette anschließend auf die Ebene im Lager, in der das Lagerfach reserviert wurde, transportiert. Nach einer Zwischenlagerung im sog. Liftübergabefach übernimmt ein schienengeführtes, führerloses Fahrzeug (Manitrac) den Transport zum endgültigen Lagerfach. Manitrac ist ein Produkt der Fa. Manitec Consulting AG, Emmen Schweiz, die auch das Lagerkonzept erstellt hat. Auslagerungsvorgänge erfolgen analog zu den Einlagerungen. Da Ein- und Auslagerungen über das gleiche Liftübergabefach abzuwickeln sind, ist eine Synchronisation dieser Vorgänge erforderlich.

Die Manitracs können sich in einem der beiden Blöcke auf jeweils einer Ebene frei bewegen und

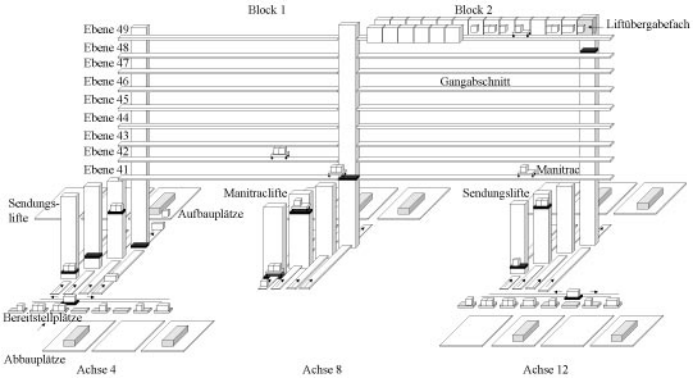

Abb. 4. Layout des Hochregallagers

Verladeaktionen durchführen. Um die Ebene oder den Block in einen neuen, sog. *Gangabschnitt* zu wechseln, muß ein Manitrac den Manitraclift der zugeordneten Gasse benutzen (vgl. Abb. 3). Der Manitraclift muß auch genutzt werden, um Einlagerungen oder Auslagerungen über die mittlere Achse durchzuführen, die dort nur in Verbindung mit einem Manitrac möglich sind.

Zur Steuerung des Lagers existieren neben der Steuerung des Materialflusses (von Paletten und Behältern) auf Förderelementen, die durch SPS- bzw. Industrie-PC-Systeme realisiert wird, zusätzliche überlagernde Steuerungsmodule für die Disposition der Lifte und der Manitracs, die Lagerplatzvergabe und eine Datenbank, die den Anlagenzustand (Bestände, Reservierungen, Aufträge etc.) widerspiegelt (vgl. Abb. 5). Das Steuerungssystem des Warenumschlaglagers setzt sich aus dem Sendungsleitrechner (SLR) und dem Bereichsrechner (BR) zusammen. Das Teilsystem SLR stellt die Leitebene des Sendungssystems dar und besitzt Schnittstellen zum Bediener sowie zum Betreiber-Host. Hauptaufgabe ist es, Ein- oder Auslagerungsaufträge vom Host oder vom Bediener entgegenzunehmen und an das Teilsystem BR weiterzuleiten. Neben Lagerverwaltungsaufgaben übernimmt das Teilsystem SLR die Archivierung und stellt Statistiken und Dialoge für administrative und Materialflussaufgaben zur Verfügung.

Die Hauptfunktion des BR ist, für das Sendungssystem den Materialfluß, der sich aus Palettenein- und -auslagerungen sowie aus Vorgaben des SLR ergibt, unter Berücksichtigung der Förderstrategien abzuwickeln, die in den Modulen *MT_Dispo*, *ML_Dispo* und *SL_Dispo* (zur Steuerung der Manitracs und der Lifte) und der Lagerplatzvergabe hinterlegt sind. Neben den zu realisierenden Materialflußfunktionen gibt es Funktionen zur Anlagenzu-

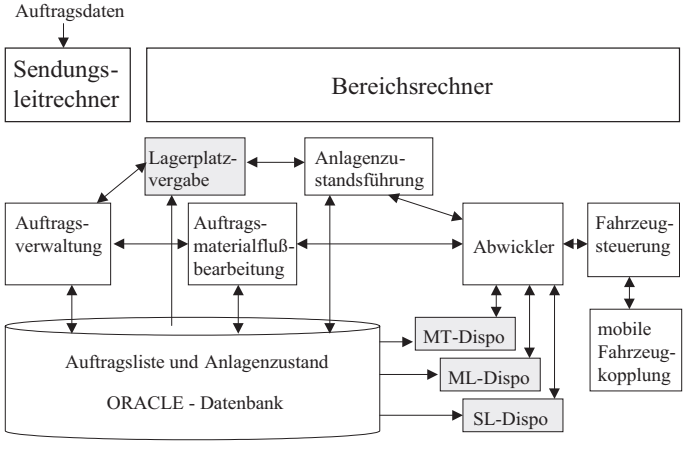

Abb. 5. Architektur des Sendungssystems

standsführung (die alleinige Schnittstelle zur Datenbank) sowie Sicherheitsfunktionen für einen gesicherten Zutritt zum Lager und definierten Sicherheitsbereichen und Verhaltensregeln im Brandfall.

Über eine auf Funktionsgruppen (Lifte, Manitracs etc.) bezogene Ereignis-Zustands-Matrix werden vom Abwickler Aktionen ausgelöst und der Materialfluß kontrolliert. Zur Kommunikation zwischen allen Systemkomponenten wird die Kommunikationssoftware PROCOM eingesetzt, ein Produkt der ALSTOM AT, das eine auf „Mailboxen" und TCP/IP basierende Schnittstelle zwischen Prozessen zur Verfügung stellt.

Die Fahrzeugsteuerung setzt Aufträge für mobile Fahrzeuge in fahrzeugspezifische Steuersequenzen um. Die mobile Fahrzeuge-TCP/IP-Kopplung hat die Aufgabe, Nachrichten zwischen den mobilen Fahrzeugen und dem BR auszutauschen.

Ziel der Kopplung des Steuerungssystems mit dem Simulationsmodell war es, die in Abb. 5 grau hinterlegten Module der Software jeweils einzeln und in Kombination durch Kopplung an das Simulationsmodell im Rahmen von Back-to-back-Tests zu überprüfen.

Die Tests beziehen sich allein auf den Normalbetrieb, d. h. Störungen werden zunächst nicht betrachtet. Die Darstellung des Umsystems (Lager und Fahrzeuge) und die Generierung der Ein- und Auslagerungsaufträge verbleibt in allen Phasen der Tests beim Simulationsmodell. Es werden also alle Testfälle von den übergeordneten Auftragsdaten, die im Simulationsmodell erzeugt werden, abgeleitet. Die entsprechende Funktionalität des Betreiber-Hosts obliegt somit allein dem Simulationsmodell.

4.2 Realisierung der Kopplung

Drei Problembereiche sind im Rahmen der Kopplung tangiert. Im Unterschied zur Steuerungssoftware erfolgt die Kommunikation im Simulationsmodell zwischen den Systemkomponenten durch Funktionsaufrufe. Hat z. B. ein Manitrac einen Transportauftrag erledigt, folgt ein Aufruf der Manitrac-Disposition, die dem Fahrzeug als Ergebnis einen neuen Auftrag zurückgibt. Dieser wird dann vom Manitrac abgearbeitet. Die Steuerungssoftware sendet ein Telegramm an das Modul *MT_Dispo*, dessen Antworttelegramm dann verarbeitet wird. Die Kopplung erfordert eine geeignete Anpassung der Kommunikation im Simulationsmodell.

Ferner sind zur Realisierung der Kommunikation zwischen den Systemen für alle Komponenten, die durch Telegramme angesprochen werden, Konvertierungsfunktionen für die Namen bereitzustellen. Die Lagerplätze sind z. B. im Simulationsmodell durch einen Objektnamen, im realen System aber durch eine andere Kodierung bezeichnet.

Zur technischen Ausgestaltung der Schnittstellen sind die Einführung von TCP/IP-Sockets[2] und Implementierung eines Übersetzers der ein- und ausgehenden Nachrichten sowie Methoden zur weiteren Verarbeitung eingehender Nachrichten im Simulationsmodell notwendig. Der gesamte Telegrammverkehr wird über ein zentrales Objekt mit verschiedenen Methoden realisiert. Alle Telegramminhalte orientieren sich am Telegrammverkehr der Steuerungssoftware und werden im Simulationsmodell für die interne Verarbeitung entsprechend übersetzt. Zur Kopplung ist es auf Seiten der Steuerungssoftware lediglich notwendig, die Telegramme an eine andere Adresse (das Modell) zu schicken.

Ein Problem der Kopplung beruht darauf, daß SiMPLE ++ keine Binär-Daten als Bestandteil eines Telegramms verarbeiten kann. Dies hat zur Folge, daß die Telegrammstruktur leicht verändert werden muß, um die Binärdaten in ASCII-Text umzuwandeln.

Ein größeres Problem stellt das Anhalten des Simulationsmodells dar. Sobald ein Anforderungstelegramm abgeschickt wird, muß das Simulationsexperiment zur Vergleichbarkeit mit den Ergebnissen der „reinen" Simulation und zur Vermeidung von Synchronisationsproblemen im Anlagenzustand so lange unterbrochen werden, bis das Antworttelegramm angekommen ist (das Modell läuft deutlich schneller als Realzeit). Unter SiMPLE ++ besteht zwar die Möglichkeit, die Simulation anzuhalten, die Simulation wird aber automatisch fortgesetzt, sobald irgendein Telegramm empfangen wird. Um dieses Verhalten zu unterbinden, sind weitere Anpassungen erforderlich.

Ein weiterer wesentlicher Unterschied zum realen System ist die Darstellung des Anlagenzustands.

[2] SiMPLE ++ stellt einen Socket als Objektbaustein zur Verfügung. Ein- und ausgehende Nachrichten müssen hier die Form von Strings (ASCII-Text) besitzen. Wird ein Telegramm an den Socket geschickt, so wird automatisch eine Methode, die am Socket eingetragen werden kann, ausgelöst. Mittels dieser Methode können dann eingehende Nachrichten verarbeitet werden.

Die Verwaltung des Anlagenzustands erfolgt mittels der vom Simulator SiMPLE + + angebotenen Informationsobjekte (Tabellen, globale Variablen, Objektattribute etc.). Diese werden nicht zentral in einer Datenbank oder einem zentralen Objekt geführt, sondern zu einem Großteil dezentral. So werden z. B. Statusinformationen über die Manitracs an diesen selbst als Attribut geführt.

Zur Synchronisation der Anlagenzustandsführung ist es wie beschrieben entweder notwendig, daß auch das Simulationsmodell nur auf die Datenbank zugreift (und alle bisherigen internen Anlagenzustandsführungen durch entsprechende Datenbankzugriffe ersetzt werden) oder diese Änderungen dem Simulationsmodell mitgeteilt werden, und dieses die Änderungen in der eigenen (verteilten) Anlagenzustandsführung nachvollzieht. In diesem Projekt ist eine doppelte Anlagenzustandsführung realisiert, indem die Funktion des Moduls *Anlagenzustandsführung* nunmehr vom Simulationsmodell ausgeführt wird, d. h. die entsprechenden Telegramme werden für die Tests an das Modell umgeleitet. Das Simulationsmodell führt die schreibenden Zugriffe auf der Datenbank per ODBC-Schnittstelle aus und veranlaßt gleichzeitig die Modifikation der internen Anlagenzustandsführung.

Wird innerhalb des Simulationsmodells eine Zustandsänderung (dezentral) vollzogen, so ist es erforderlich, auch an diesen Stellen im Simulationsmodell entsprechende Datenbankzugriffe durchzuführen. Hierzu werden die Methoden für die unterschiedlichen Datenbankzugriffe in einem Objekt zusammengefasst. Neben Zugriffen zur Initialisierung werden Platzreservierungen, Gangabschnittsreservierungen etc. in der Datenbank protokolliert.

Im Simulationsmodell werden in allen Testkonstellationen zunächst die Steuerungsmodule des Modells aufgerufen. Hier wird geprüft, ob das entsprechende Modul der Steuerungssoftware getestet werden soll. Ist das der Fall, so wird das Anforderungstelegramm durch Aufruf einer zentralen Methode ausgelöst. Sobald das Antworttelegramm eingetroffen ist, wird das Ergebnis dem aufrufenden Objekt mitgeteilt und die Simulation fortgeführt.

4.3 Testaufbau

Die Tests der Steuerungssoftware erfolgten in zwei Projektschritten. Diese Aufteilung liegt darin begründet, daß eine frühzeitige Teilnutzung des Lagers geplant wurde, wobei auch die Steuerungssoftware nur zum Teil implementiert wurde. Im ersten Schritt wurden die Module *Lagerplatzvergabe*, *SL_Dispo* und Teile der Manitracdisposition getestet. Da in der Teilnutzung die Manitraclifte nur zum Umsetzen der Manitracs, nicht aber für Ein- und Auslagerungsvorgänge an der Achse 8 eingesetzt wurden, wurde auf die vollständige Implementierung und zwangsläufig auch auf den Test dieser Funktionalitäten im ersten Schritt verzichtet.

Es wurden zunächst die Basisfunktionalitäten des Telegrammverkehrs und der Datenbank getestet. Hier wurden bereits erste Fehler aufgedeckt. Diese betrafen zum einen die Telegramm- und Datenbankstrukturen, zum anderen aber auch ein Modul, das bereits vor dem Projekt als anwendbares Teilprogramm bestand und in anderen Systemen im Einsatz ist. Das Auffinden dieses Fehlers wurde als besonders wertvoll erachtet.

Es wurde dann mit dem Test des Moduls *SL_Dispo* begonnen. Hier wurden exakt die gleichen Experimente wie zuvor in der Planungssimulation durchgeführt, d. h. es konnte auch der Systemdurchsatz weiterhin beurteilt und mit alten Ergebnissen verglichen werden.

Nach erfolgreichen Tests (d. h. es wurden diverse Fehler aufgedeckt) folgten die Versuche für die Lagerplatzvergabe und die erforderlichen Teile der Manitracdisposition für die Teilnutzung des Lagers. Die Testaktivitäten wurden dann unterbrochen, um die bestehenden Module vor Ort in Betrieb zu nehmen.

Nachdem die Teilinbetriebnahme erfolgt war, wurde auf Seiten der Steuerungssoftware mit der Implementierung weiterer Funktionalitäten begonnen und die Tests für die Module *MT_Dispo* und die *ML-Dispo* weitergeführt.

Im letzten Schritt der Integrationstests wurden alle Entscheidungen vom SLR übernommen, wobei das Simulationsmodell nur noch die Physik der Anlage abbildete und die Erzeugung der Testdaten übernahm.

5 Bewertung und Ausblick

Insgesamt kann von einem positiven Effekt der simulationsgestützten Tests auf die Inbetriebnahmezeit vor Ort gesprochen werden. So wurden viele Fehler bei den Back-to-back-Tests aufgedeckt, die sich beim Test an der realen Anlage z. T. erst durch

Hardware-mäßig installierte Sicherheitsabschaltungen bemerkbar gemacht hätten. Es bleibt also anzumerken, daß solche Black-box-Tests an der Anlage in diesem Fall sogar gefährlich sind und weitere Tests aufgrund der möglichen Schäden an der Anlage um Wochen verzögert hätten. Auf eine ausführliche Darstellung der Fehler soll hier verzichtet werden.

Auf Seiten der Simulation wurden für alle Änderungen und die Unterstützung bei den Tests etwa zwei Mannmonate benötigt. Ein wesentlicher Anteil fiel auf die Anpassung des Modells und nur etwa sechs Manntage direkt auf die Unterstützung der eigentlichen Tests. Der Nutzen des Projekts soll kurz am Beispiel der Tests der Sendungsliftdisposition dargestellt werden. Hier konnte die Teilnutzung gegenüber dem alten Plan nach nur 25 % der Zeit (etwa vier Tage statt zwei Wochen), und insgesamt mit nur etwa 80 % des Aufwands realisiert werden, wobei wir davon ausgehen, daß bei den Tests gegen das Simulationsmodell der Simulationsdienstleister ebenfalls insgesamt fünf Manntage für die Anbindung der Sendungsliftdisposition und die Unterstützung dieser Modultests investiert hat.

Ähnliche Ergebnisse lassen sich für die anderen Module angeben, wobei eine exakte Quantifizierung des Nutzens nicht möglich ist. Es kann aber davon ausgegangen werden, daß aufgrund der deutlich höheren Anzahl der Tests auch die Güte der Software höher zu bewerten ist, wobei der Aufwand insgesamt noch unter dem des traditionellen Testens (ohne Simulationsmodell) liegen sollte. Wesentlich ist aber, daß die Zeit bis zum tatsächlichen Betrieb der Anlage drastisch reduziert werden konnte. Nach Einschätzung der ALSTOM AT GmbH konnte die Inbetriebnahmezeit vor Ort insgesamt um etwa einen Monat verkürzt werden, wobei drei für tatsächliche Tests am Simulationsmodell genutzte Tage einer Reduktion der Inbetriebnahmezeit um zwei Wochen entsprechen.[3]

Nach einem solchen Projekt stellen sich Fragen zur Standardisierung und zur Verbesserung der Vorgehensweise. Zunächst ist festzuhalten, daß es sinnvoll ist, den Simulationsdienstleister bereits vor Beginn einer Simulationsstudie, wenn diese neben der Kopplung auch der Systembewertung dienen soll, in die Standards der später zu nutzenden Kommunikation einzuweisen und mit ihm die (grobe) Architektur des Simulationsmodells auf Realisierbarkeit und Sinnhaftigkeit zu überprüfen. Eine solche Vorgehensweise ist von eminenter Bedeutung und hat sich im vorgestellten Beispiel bewährt.

Der Simulationsdienstleister sollte auf der anderen Seite in die Planung und Konzeption des Steuerungssystems einbezogen werden. So können Schwierigkeiten, wie sie durch die Problematik der Binärdaten im Projekt entstanden, durch eine koordinierte Planung umgangen werden.

Die Schnittstellen zu externen Systemen sollten im Rahmen des Telegrammverkehrs standardisiert werden. So können Bausteine definiert werden, die ein Anhalten der Simulation bis zum Eintreffen eines Antworttelegramms automatisieren, da eine solche Kommunikationsstruktur typisch ist. Ein Nebenprodukt dieses Projekts stellt ein (spezieller) Baustein mit einer solchen Funktionalität dar.

Das größte Verbesserungspotential für die (nachträgliche) Kopplung bietet sich im Bereich der Anlagenzustandsführung. Die angewandte Praxis, alle schreibenden Zugriffe von nur einem Modul (hier innerhalb des Simulationsmodells) durchführen zu lassen, hat sich im Projekt prinzipiell zwar als erfolgreich erwiesen, eine spätere Kopplung kann aber wesentlich erleichtert werden: Im Simulationsmodell sollte eine eigene Datenbank unter Nutzung der vom Simulationswerkzeug angebotenen Datenstrukturen oder unter Verwendung von Datenbanksystemen, auf die permanent vom Simulationsrechner zugegriffen werden kann, angelegt werden. Unter SiMPLE ++ entspräche dies z. B. einem Modul mit Tabellen und den entsprechenden Zugriffsmethoden oder einer Access-Datenbank. Bei einer Kopplung ist es dann nicht notwendig, den vollständigen Programmcode nach den Stellen zu durchsuchen, an denen Statusänderungen an das andere System gemeldet werden müssen. Es reicht in diesem Fall aus, die Zugriffsmethoden der zentralen Instanz zur Verwaltung des Anlagenzustands zu erweitern.

Auch die Veränderung der modellinternen Zustände, die vom Steuerungssystem z. B. per Telegramm gemeldet werden, wird durch eine solche Vorgehensweise deutlich vereinfacht. Es müssen

[3] Es bleibt anzumerken, daß die Geschwindigkeit zur Durchführung der Experimente im wesentlichen durch die Datenbankzugriffe bestimmt wird, jedoch deutlich schneller als Realzeit simuliert werden kann. Im realen System wird (neben der Datenbank) ein gemeinsamer Speicher („shared memory") zur Beschleunigung der Zugriffszeiten eingesetzt.

dann nur die entsprechenden Methoden, die sonst aus dem Simulationsmodell heraus aufgerufen werden, ausgeführt werden.

Das vorgestellte Projekt ist ein Beispiel dafür, daß der durchgängige Einsatz der Simulation über die Konzeptplanung von Lagersystemen hinaus weitere Synergieeffekte freisetzen und die Inbetriebnahmezeit solch komplexer Systeme deutlich reduzieren kann. Unter Einbeziehung der im Projekt erkannten Verbesserungspotentiale ist ein Rahmenkonzept für die durchgängige Nutzung eines Simulationsmodells von der Konzeptstudie von Lägern über Modultests bis zu Systemtests für Projekte der gleichen Art entwickelt worden. Die Arbeit sollte dabei aufgezeigt haben, wie sich Modultests mittels bestehender Simulationsmodelle im Rahmen einer iterativen Vorgehensweise durchführen und dabei die Vorteile der Bottom-up- und Top-down-Testansätze auf elegante Weise verbinden lassen.

Literatur

1. Boehm, B.W.: Guidelines for verifying and validating software requirements and design specification. Euro IFIP, 711–719 (1979)
2. Boehm, B.W.: Industrial software metrics top 10 list. IEEE Software 4 (5), 84–85 (1987)
3. Breilmann, E., Herrmann, H.: So schnell kann's gehen. Commissioning by Simulation – Test des New World Cargo Centres am Bildschirm. Materialfluß (10–12), 12–15 (1997)
4. Bröhl, A.-P. , Dröschel, W. (Hrsg.): Das V-Modell. Der Standard für die Softwareentwicklung mit Praxisleitfaden. 2. Aufl. München: Oldenbourg 1995
5. Brooks, F.P.: The Mythical Man Month. Reading: Addison-Wesley 1995
6. Dröschel, W., Heuser, W., Midderhoff, R. (Hrsg.): Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97. München: Oldenbourg 1998
7. Gutenschwager, K., Fauth, K.A.: Über die Simulation zum funktionalen System. Fördern und Heben 48 (6), 430–432 (1998)
8. Jünemann, R.: Materialfluß und Logistik. Berlin Heidelberg New York Tokyo: Springer 1989
9. Jünemann, R., Beyer, A.: Steuerung von Materialfluß- und Logistiksystemen. 2. Auflage, Berlin Heidelberg New York Tokyo: Springer 1998
10. Klibi, K.: Simulation zur Optimierung der SPS-Softwareentwicklung. White Paper, Dortmund: SimulationsDienstleistungsZentrum GmbH 1999
11. Kosturiak, J., Gregor, M.: Simulation von Produktionssystemen. Berlin Heidelberg New York Tokyo: Springer 1995
12. Law, A.M., Kelton, W.D.: Simulation Modeling & Analysis. 2. Auflage, New York: McGraw-Hill 1991
13. Liggesmeyer, P.: Modultest und Modulverifikation. Mannheim: BI Wissenschaftsverlag 1990
14. Liggesmeyer, P., Rothfelder, M., Rettelbach, M., Ackermann, T.: Qualitätssicherung Software-basierter technischer Systeme – Problembereiche und Lösungsansätze. Informatik Spektrum 21, 249–258 (1998)
15. Myers, G.J.: Methodisches Testen von Programmen. 6. Auflage, München: Oldenbourg 1999
16. Noche, B.: Kopplung von Simulationsmodellen mit Leitrechnern. White Paper, Dortmund: SimulationsDienstleistungsZentrum GmbH 1999
17. Pagel, B.-U., Six, H.-W.: Software Engineering. Band 1: Die Phasen der Softwareentwicklung. Reading: Addison-Wesley 1994
18. Pidd, M.: Computer Simulation in Management Science. 3. Auflage. Chichester: Wiley 1992
19. Royce, W.: Software Project Management. A Unified Framework. Reading: Addison-Wesley 1998
20. Sommerville, I.: Software Engineering. 4. Auflage. Reading: Addison-Wesley 1992
21. Spieckermann, S., Voß, S., Wortmann, D.: Praxisorientierte Klassifikationsmerkmale für ereignisorientierte Simulationswerkzeuge. Zeitschrift für Planung 8, 81–97 (1997)
22. Spieckermann, S., Gutenschwager, K.: Discrete event simulation in the automotive industry: Applications and prospects. In: Heller, M. (Hrsg.): Automotive Simulation. 5th European Engineering Simulation Symposium (S.65–74). San Diego: Society for Computer Simulation 1997
23. Swain, J.J.: Simulation goes mainstream. OR/MS Today 24 (5), 35–46 (1997)
24. VDI: Simulation von Logistik-, Materialfluß- und Produktionssystemen. VDI-Richtlinie 3633. Düsseldorf: VDI-Verlag 1993
25. VDI: Logistikleitstand für die Fabrik. VDI-Richtlinie 4410. Berlin: Beuth 1995
26. Wloka, J., Spieckermann, S.: Warenumschlag. In: A. Kuhn, M. Rabe (Hrsg.): Simulation in Produktion und Logistik. Berlin Heidelberg New York Tokyo: Springer 1998, S.11–34