Die Spezifikationssprachen MSC und SDL Teil 1: Message Sequence Chart (MSC)

The Specification Languages MSC and SDL Part 1: Message Sequence Chart (MSC)

Jens Grabowski, Ekkart Rudolph und Michael Schmitt

Im Telekommunikationsbereich haben sich *Message Sequence Chart* (MSC) und *Specification and Description Language* (SDL) als wichtigste Modellierungssprachen etabliert. Dieser zweiteilige Artikel beschreibt die Konzepte und den Einsatz der beiden Sprachen. Im vorliegenden ersten Teil wird ein Überblick über SDL und MSC gegeben und die Anforderungsspezifikation mit Hilfe von MSC beschrieben.

In the telecommunication area, *Message Sequence Chart* (MSC) and *Specification and Description Language* (SDL) are the dominating modeling languages. This two-part article describes concepts and usage of both languages. In this first part, an overview of SDL and MSC is given and the specification of requirements with MSC is described.

1 Einführung in MSC und SDL

Im Gegensatz zu weiten Teilen der Informatik besteht in der Telekommunikation seit langem die Notwendigkeit, herstellerunabhängige und präzise Standards für Kommunikationsprotokolle und -dienste zu erstellen. Diese Anforderung hat dazu geführt, dass man sich schon frühzeitig mit der Entwicklung der formalen Modellierungssprachen *Message Sequence Chart* (MSC) und *Specification and Description Language* (SDL) beschäftigt hat. Im Gegensatz etwa zur *Unified Modeling Language* (UML) [1] liegt MSC und SDL eine formale Semantik zugrunde, die die Bedeutung der einzelnen Sprachkonstrukte klar und eindeutig definiert.

MSC und SDL sind von der *International Telecommunication Union* standardisiert worden. Für beide Sprachen existieren eine graphische Repräsentation (MSC/GR bzw. SDL/GR) für die Benutzung durch Menschen und eine textuelle Notation (MSC/PR bzw. SDL/PR) für eine maschinelle Weiterbearbeitung. MSC und SDL werden häufig gemeinsam im Software-Entwicklungsprozess eingesetzt. MSC dient dabei zur Anforderungsdefinition, Testfallspezifikation und Dokumentation, während SDL in der Spezifikationsphase und der Implementierungsphase eingesetzt wird.

2 Die Message Sequence Chart Sprache

MSC ist eine vom ITU-T als Recommendation Z.120 [7] standardisierte graphische Spezifikationssprache. Sie dient zur Beschreibung des Kommunikationsverhaltens zwischen Systemkomponenten und deren Umgebung. Die Kommunikation wird durch den Austausch von Nachrichten (*Messages*) spezifiziert.

Die MSC-Sprache hat sich aus den *OSI Time Sequence Diagrams* [2; 6] zu einer vollständigen graphischen Sprache mit formaler Syntax und Semantik entwickelt [7; 8]. Mit den *Sequence Diagrams* wurde eine Variante von MSC in die UML integriert [1]. Obwohl beide Sprachen auf den gleichen Prinzipien beruhen, gibt es auf Grund der verschiedenen Anwendungsbereiche unterschiedliche Sprachkonstrukte und Unterschiede in der Darstellung. Durch eine Kombination der Stärken von MSC und den Sequence Diagrams bemüht man sich zur Zeit verstärkt um eine Harmonisierung der beiden Diagrammarten [3; 5].

In diesem Artikel werden die wichtigsten Konstrukte der MSC-Sprache an Hand von MSCs[1] für das in der Einlei-

[1] Nachfolgend wird die Abkürzung MSC sowohl zur Bezeichnung der MSC-Sprache als auch zur Bezeichnung von Diagrammen in der MSC-Sprache verwendet.

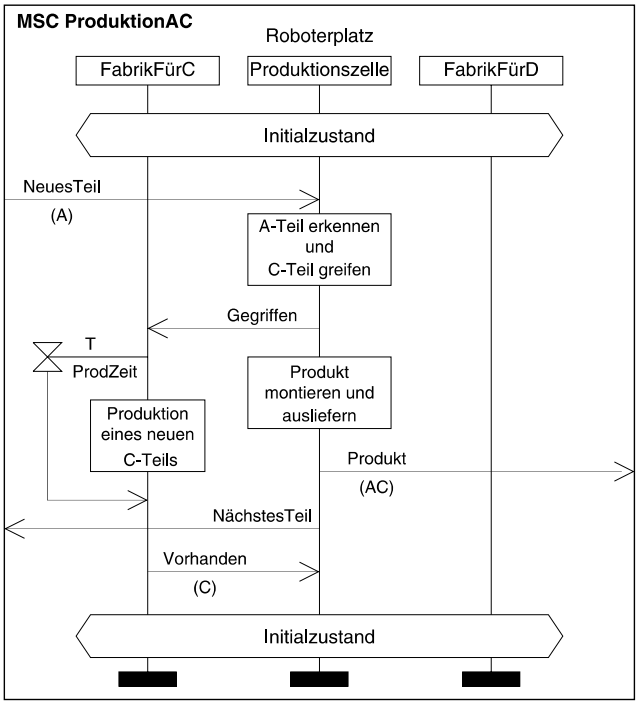

Bild 1: Ein Basic-MSC.

tung zu dieser Artikelreihe beschriebene Beispiel erläutert [4]. Das Beispiel beschreibt einen Produktionsprozess, in dem eine Produktionszelle aus vier verschiedenen Einzelteilen vom Typ A, B, C und D zwei Produkte vom Typ AC und BDC herstellt.

2.1 Basic-MSC

Basic-MSC umfasst alle Sprachkonstrukte, die notwendig sind, um den Nachrichtenfluss zu spezifizieren. Diese Sprachkonstrukte sind: *Instanz*, *Message*, *Environment*, *Action*, *Timer-Start*, *Timeout*, *Timer-Stop*, *Create*, *Stop* und *Condition*.

Ein Basic-MSC mit dem Namen `ProduktionAC` ist in Bild 1 gezeigt. Es beschreibt die Produktion eines aus einem A- und einem C-Teil zusammengesetzten Produkts. Das MSC abstrahiert von den Details und zeigt nur den Informationsaustausch zwischen den drei Hauptkomponenten des Produktionsprozesses: Der Produktionszelle und den zwei Fabriken für C- und D-Teile.

Der Produktionsprozess befindet sich zu Beginn in seinem Initialzustand: Das System ist initialisiert und Teile der Typen C und D sind gefertigt, eingelagert und der Produktionszelle zur Verfügung gestellt worden. Die Systemumgebung übergibt ein A-Teil an die Produktionszelle. Diese erkennt den Typ des Teils, greift das für die Produktion notwendige C-Teil und meldet das Entfernen des C-Teils vom Greifplatz an die Fabrik, die C-Teile herstellt. Ein Produkt vom Typ AC wird gefertigt und dann ausgegeben. Danach meldet die Produktionszelle die Bereitschaft für die Bearbeitung des nächsten A- oder B-Teils an die Systemumgebung. Parallel zu den Aktionen der Produktionszelle wird ein neues C-Teil produziert und zur Verfügung gestellt.

Instanz und Message

Die wichtigsten Sprachkonstrukte von Basic-MSC sind *Instanz* und *Message*. Instanzen sind Komponenten, die untereinander oder mit der Systemumgebung asynchron Messages austauschen können.

In der graphischen Form werden Instanzen als vertikale Linien oder alternativ als Säulen dargestellt. Innerhalb des Instanzkopfes wird der Instanzname spezifiziert, zusätzlich kann ein Typ angegeben werden (z. B. Instanz `Produktionszelle` vom Typ `Roboterplatz` in Bild 1). Das Ende einer Instanz kann durch ein *Instanz-Ende-Symbol* (▬) beschrieben werden. Ein Instanz-Ende bedeutet nicht, dass die Instanz gestoppt ist, sondern lediglich dass in dem MSC keine weiteren Ereignisse für die Instanz beschrieben werden.

Messages werden durch Pfeile dargestellt. Diese Pfeile können horizontal oder geneigt sein, um den Zeitverlauf anzuzeigen. Eine Message definiert zwei Ereignisse: Der Pfeilanfang beschreibt das Senden und die Pfeilspitze bezeichnet die Verarbeitung einer Message. Die Beschriftung einer Message besteht aus ihrem Namen und optionalen Message-Parametern, die in Klammern angegeben werden müssen (z. B. Message `NeuesTeil` mit dem Parameter A in Bild 1).

Entlang jeder Instanzachse wird eine Totalordnung der spezifizierten Message-Sende- und Message-Verarbeitungsereignisse angenommen. Ereignisse auf verschiedenen Instanzachsen werden durch Message-Kommunikation partiell geordnet, da eine Message zuerst gesendet werden muss, bevor sie verarbeitet werden kann.

Environment

Die Diagrammfläche eines MSC wird durch einen rechteckigen Rahmen begrenzt. Der Diagrammrahmen definiert die Systemumgebung und heißt *Environment*. Messages, die aus der Systemumgebung kommen oder an die Systemumgebung gesendet werden, beginnen und enden auf dem Environment (z. B. die Messages `NeuesTeil` und `Produkt` in Bild 1). Im Gegensatz zur Totalordnung entlang der Instanzachsen ist für Sende- und Verarbeitungsereignisse auf dem Environment keine Ordnung definiert.

Action

Zusätzlich zur Message-Kommunikation können Aktionen von Instanzen in Form von *Actions* spezifiziert werden. Eine Action wird durch ein Rechtecksymbol dargestellt, das beliebigen Text enthalten kann (z. B. Action `‚Produkt montieren und ausliefern‘` in Bild 1).

Timer

Zur Beschreibung von Timern bietet MSC die Sprachkonstrukte *Timer-Start*, *Timeout* und *Timer-Stop* an. Timer-Start spezifiziert das Setzen, Timeout den Ablauf und Timer-Stop das Zurücksetzen eines Timers. In MSC ist ein Timer

immer einer Instanz zugeordnet. Die graphischen Timer-Symbole sind daher immer mit der dem Timer zugeordneten Instanz verbunden.

Ein Timer-Start wird durch ein mit der Instanzachse verbundenes Sanduhr-Symbol dargestellt. Ein zugehöriges Timeout wird durch einen Pfeil beschrieben, der am Sanduhr-Symbol beginnt und auf der Instanzachse endet. Ein Timer-Stop wird durch ein mit der Instanzachse verbundenes Kreuz beschrieben. Ein Timersymbol wird mit dem Timernamen und optional mit Timer-Parametern versehen. Ein Timer-Start und das zugehörige Timeout werden in Bild 1 verwendet, um die Produktionszeit für ein Teil des Typs `C` zu modellieren. Der Timer hat den Namen `T` und den Parameter `Prodzeit`, der die Laufzeit des Timers spezifiziert.

Condition

Eine *Condition* beschreibt einen Zustand, der sich auf eine Menge der im MSC enthaltenen Instanzen bezieht. Graphisch werden Conditions durch Sechsecke dargestellt, die die Instanzen, auf die sich die Condition bezieht, überdecken. Conditions werden zur Beschreibung von wichtigen Systemzuständen benutzt. In Bild 1 befinden sich zwei Conditions, die beide den globalen Systemzustand `Initialzustand` beschreiben.

Create und Stop

Die MSC-Sprache enthält die Konstrukte *Create* und *Stop* für die dynamische Erzeugung und Terminierung von Instanzen. Ein Create wird durch einen gestrichelten Pfeil mit optionalen Parametern beschrieben. Ein Create-Pfeil beginnt an der Erzeuger-Instanz und endet am Kopf der erzeugten Instanz. Eine Instanz kann sich selbst durch eine Stop-Aktion terminieren. Ein Stop wird graphisch durch ein Kreuz am Ende der Instanzachse spezifiziert.

2.2 Strukturelle Sprachkonstrukte

Strukturelle MSC-Sprachkonstrukte bezeichnen Sprachelemente, die über die Beschreibung des reinen Messageflusses hinausgehen. Mit ihnen lassen sich MSCs und MSC-Teile zu komplexeren Abläufen kombinieren (*Inline-Expressions* und *High-Level-MSC*), MSC-Diagramme in anderen MSC-Diagrammen wiederverwenden (*References*), MSC-Instanzen verfeinern (*Decomposition*) und allgemeine Ereignisstrukturen für Instanzen definieren (*Coregion* und *General Ordering*). Aus Platzgründen kann im Rahmen dieses Artikels nur auf *Inline-Expressions*, *References* und *High-Level-MSC* (HMSC) eingegangen werden.

Inline-Expressions

Mit *Inline-Expressions* können Teilabläufe, die innerhalb eines MSC-Diagramms spezifiziert worden sind, zu komplexeren Abläufen kombiniert werden. Für die Kombination bietet MSC die Operatoren *alt*, *par*, *loop*, *opt* und *exc*

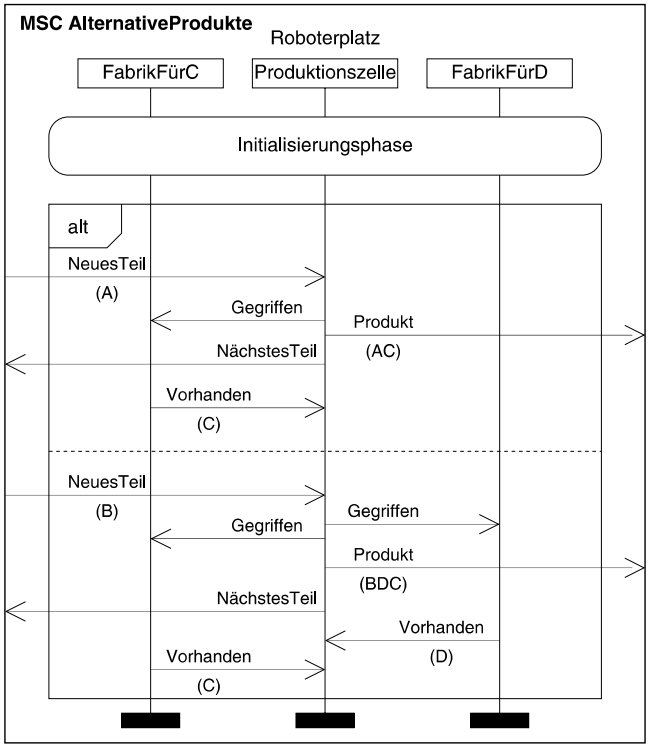

Bild 2: MSC mit strukturellen Sprachkonstrukten.

an. Sie erlauben es, die Wiederholung von Teilabläufen (*loop*-Operator), alternative Teilabläufe (*alt*-Operator), die parallele Komposition von Teilabläufen (*par*-Operator), optionale Teilabläufe (*opt*-Operator) und Ausnahmen in Form von Teilabläufen (*exc*-Operator) zu spezifizieren. Graphisch werden Inline-Expressions als Rechtecke mit gestrichelten Linien als Separatoren für Teilabläufe dargestellt. Der Operator wird in der linken oberen Ecke spezifiziert.

Eine Inline-Expression mit einem *alt*-Operator befindet sich in Bild 2. Die alternativen Teilabläufe beschreiben die Fertigung von Produkten der Typen `AC` und `BDC` in Abhängigkeit des von der Systemumgebung übergebenen Basisteils (`A` oder `B`).

References

References ermöglichen es, MSCs in anderen MSCs wieder zu verwenden. Eine Reference referenziert ein anderes MSC über dessen Namen, d. h. eine Reference kann als Platzhalter für das referenzierte MSC angesehen werden. Graphisch werden References durch ein Rechteck mit abgerundeten Ecken dargestellt. Eine Reference befindet sich auch in Bild 2. Sie referenziert ein MSC mit dem Namen `Initialisierungsphase`, das die Initialisierung des Produktionsprozesses beschreibt.

High-Level-MSC

High-Level-MSC (HMSC) erlaubt es, die Kombination von MSCs in Form eines gerichteten Graphen zu beschreiben. Die Knoten eines HMSC-Diagramms sind ein Anfangsknoten (▽), Endknoten (△), Konnektoren (○), References und

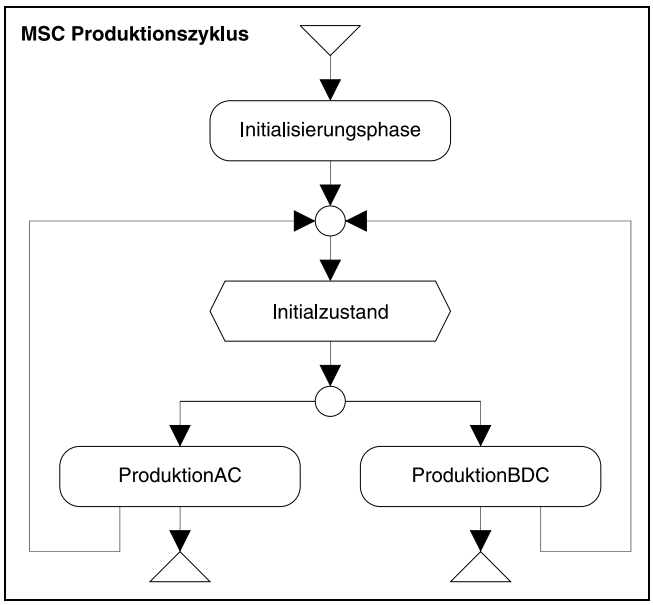

Bild 3: Ein High-Level-MSC.

Conditions. In HMSC konzentriert man sich auf die Darstellung der Kombination von MSC-Diagrammen und abstrahiert von den Instanzen und dem Messagefluss. HMSC-Diagramme werden häufig auch *Roadmaps* genannt. Mit ihnen lässt sich die sequentielle, parallele und alternative Kombination von MSCs in einer sehr intuitiven Form beschreiben.

Der HMSC in Bild 3 beschreibt die erwarteten Abläufe des Produktionsprozess-Beispiels: Nach der Initialisierungsphase befindet sich der Produktionsprozess in seinem Initialzustand. Im Initialzustand können entweder Produkte vom Typ AC oder vom Typ BCD hergestellt werden. Nach der Herstellung eines Produkts befindet sich der Produktionsprozess wieder im Initialzustand und ein neues Produkt kann gefertigt werden, oder der Produktionsprozess endet.

2.3 Weitere Sprachkonstrukte

In diesem Artikel konnten aus Platzgründen nur die wichtigsten Elemente der MSC-Sprache vorgestellt werden. Neben den genannten Konstrukten enthält MSC noch einige weitergehende Konzepte, die MSC zu einer vollständigen Spezifikationssprache machen: Die zu einer Spezifikation gehörenden MSC-Diagramme können in einem *MSC-Dokument* gesammelt und strukturiert werden. MSC besitzt keine eigene Datensprache, aber eine *allgemeine Datenschnittstelle*, die es erlaubt, Datenbeschreibungen aus anderen Sprachen wie z. B. C, C++, SDL oder Java zu benutzen. Zur Spezifikation von Realzeitanforderungen können absolute Zeitpunkte und Zeitintervalle in MSC-Diagrammen spezifiziert werden. Weiterhin wurden UML-Konzepte zur objektorientierten Modellierung, wie z. B. *Control-Flow* und *Procedure-Calls*, übernommen. Nähere Informationen und eine vollständige Beschreibung der MSC-Sprache finden sich bei der SDL-Forum Society (`http://www.sdl-forum.org`).

Literatur

[1] G. Booch, J. Rumbaugh, I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[2] J. Grabowski, P. Graubmann, E. Rudolph. *The Standardization of Message Sequence Charts*. In: Tagungsband zum IEEE Software Engineering Standards Symposium 1993, IEEE, September 1993.

[3] P. Graubmann, E. Rudolph. *HyperMSCs and Sequence Diagrams for Use Case Modeling and Testing*. In: Tagungsband zur UML2000, 3rd International Conference on The Unified Modeling Language, Springer, Okt. 2000.

[4] S. Kowalewski. *Modellierungsmethoden aus der Informatik*. In: at-Automatisierungstechnik 49, Heft 9, Sept. 2001.

[5] E. Rudolph, J. Grabowski, P. Graubmann. *Towards a Harmonization of UML-Sequence Diagrams and MSC*. In: SDL'99 – The next Millenium, Elsevier, Juni 1999.

[6] ISO/IEC. *Information Technologiy – OSI Service Conventions*. ISO/IEC JTC1/SC 21 N 6341, Geneva, 1991.

[7] ITU-T SG 10. *Message Sequence Chart* (MSC). Rec. Z.120, Geneva, 2000.

[8] ITU-T SG 10. *Algebraic semantics of Message Sequence Chart*. Rec. Z.120 Annex B, Geneva, 1996.

Manuskripteingang: 19. Februar 2001.

Dr. Jens Grabowski lehrt und forscht am Institut für Telematik der Universität Lübeck. Er hat in verschiedenen Standardisierungsprojekten des European Telecommunications Standards Institute (ETSI) mitgearbeitet und war dabei unter anderem an der Entwicklung der dritten Edition von TTCN (TTCN-3) beteiligt. Seine Forschungsinteressen liegen in der Entwicklung und Anwendung von formalen Methoden für Spezifikation und Test von verteilten Systemen.

Adresse: Institut für Telematik, Universität Lübeck, Ratzeburger Allee 160, D-23538 Lübeck,
E-Mail: jens@itm.mu-luebeck.de

Dr. Ekkart Rudolph war von 1990 bis 1997 als Rapporteur und Editor für die Entwicklung und Standardisierung von MSC im ITU-T verantwortlich. Seit 1997 ist er Gastwissenschaftler in der Forschungsgruppe von Prof. Manfred Broy an der Technischen Universität München. Gegenwärtig ist er in der Weiterentwicklung von MSC engagiert, insbesondere in der Entwicklung eines MSC/UML-Testformats bei ETSI.

Adresse: Technische Universität München, Institut für Informatik, Arcisstr.21, D-80290 München,
E-Mail: rudolphe@informatik.tu-muenchen.de

Dipl.-Inform. Michael Schmitt arbeitet als wissenschaftlicher Mitarbeiter am Institut für Telematik der Universität Lübeck. Sein Forschungsschwerpunkt liegt dabei auf dem Testen von verteilten Systemen. Er ist Mitentwickler von AUTOLINK, einem Tool für die automatische Generierung von TTCN-Testsuiten basierend auf SDL-Spezifikationen und MSC-Testzweckbeschreibungen. In den Jahren 1998 bis 2000 hat er am ETSI mitgewirkt, um die Anwendung der automatischen Testfallgenerierung in der Standardisierung zu untersuchen.

Adresse: Institut für Telematik, Universität Lübeck, Ratzeburger Allee 160, D-23538 Lübeck,
E-Mail: schmitt@itm.mu-luebeck.de