CAD-Frameworks: Ergebnisse des DASSY-Projekts

Klaus Quibeldey-Cirkel

Der Wunsch des Entwicklungsingenieurs nach Werkzeugintegration ist so alt wie die Normungsgremien von ISO und DIN. Im EDA-Bereich aber scheint nun der Leidensdruck durch Ad-hoc- und Patchwork-Anpassungen groß genug, um den Ruf nach standardisierten CAD-Frameworks auch in den Etagen der Entscheidungsträger laut werden zu lassen. Worin liegen nun die Probleme „jenseits der Werkzeuge"? Wer geht sie an und wie sehen die Lösungen aus? Der folgende Beitrag gibt eine Antwort. Wir orten zunächst die Nahtstellen einer VLSI-Infrastruktur mit zentraler Datenhaltung, skizzieren anhand eines Referenzmodells die aktuelle Projekt- und Gremienlandschaft und erläutern die Framework-Konzepte, die im BMFT-Forschungsprojekt DASSY für Austauschformate und prozedurale Werkzeugschnittstellen entwickelt wurden. Die DASSY-Ergebnisse haben maßgeblich die Industriestandards der CAD-Framework-Initiative (CFI) und der EDIF-Gremien beeinflußt. DASSY: „Datentransfer und Schnittstellen für offene integrierte VLSI-Entwurfssysteme".

Kompatible Werkzeuge oder „plug in and play"

Die internationalen Bemühungen um standardisierte CAD-Integrationsrahmen, neudeutsch *Frameworks* oder gehoben *Infrastrukturen*, sollen das heterogene Spektrum an alten und neuen Werkzeugen verschiedener Hersteller breitbandig erschließen. Gemäß ihrem Kompatibilitätsanspruch wollen EDA-Anbieter und -Anwender mittelfristig ehrgeizige Ziele verwirklichen: (a) Austausch und Wiederverwendung gemeinsamer Entwurfsdaten, (b) einheitliche Benutzerführung und Methodenkontrolle sowie (c) rechnerunterstütztes Prozeß- und Ressourcen-Management.

Bild 1 zeigt das Schalenmodell einer idealisierten VLSI/CAD-Infrastruktur. Die isolierenden Schichten sollen die Entwurfskomponenten, wie Benutzerführung, Methoden, Werkzeuge und Daten, klar voneinander trennen und normative Übergänge festlegen. Die in das Zwiebelmodell getriebenen Keile symbolisieren die schnittstellenübergreifenden Dienste. Tabelle 1 ordnet wichtige Framework-Projekte und -Standardisierungsbemühungen den Schnittstellen zu.

Weltmodelle erfassen die Essenz der VLSI-Entwurfsdaten

Auf allen Darstellungsebenen und unter allen Perspektiven, d. h. den Abstraktionsebenen und Sichten im VLSI-Entwurfsraum, wird während des Entwurfsprozesses ein Informationspool angereichert, der das Entwurfsobjekt mitsamt seiner Entstehungsgeschichte beschreibt. Die Generalanforderungen an die Modellierung des Informationspools nennen wir „Weltmodell" in Anlehnung an den Begriff des „Datenmodells" aus der Datenbanktechnologie. Wie Datenmodelle, z. B. das relationale, umfassen Weltmodelle die konzeptionellen Werkzeuge zur Modellierung eines interessierenden Weltausschnitts, des öfteren *Miniwelt* genannt. Im Mittelpunkt stehen also zwei Facetten der Informationsverarbeitung: Modellieren und Schematisieren der Entwurfsinformation (Datenrepräsentation) und konsistentes Fortschreiben und Abfragen des Datenbestands (Datenverwaltung).

Vor der Implementierung steht die Spezifikation und vor der Spezifikation das „Begreifen" des Problems, das kognitive Erfassen des interessierenden Weltausschnitts. Im allgemeinen treten *konzeptionelle* Phasenbrüche im Entwicklungsprozeß auf: Die Beschreibungsmittel der Problemanalyse, der Spezifikation und der Implementierung sind in der Regel nicht kongruent. Die „semantische Kluft" zwischen ihnen ist sprichwörtlich. *Konzeptionelles Modellieren* ist das Bestreben, alle Phasen des Systementwurfs methodisch und begrifflich durchgängig zu unterstützen. Frederick Brooks [1] zählt die konzeptionelle Modellierung zu den hoffnungsvollen Kandidatinnen für eine *silver bullet* gegen ausufernde monströse Software-Projekte:

> „Fashioning complex conceptual constructs is the *essence*; *accidental* tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence."

Wie sieht nun das „conceptual construct" des

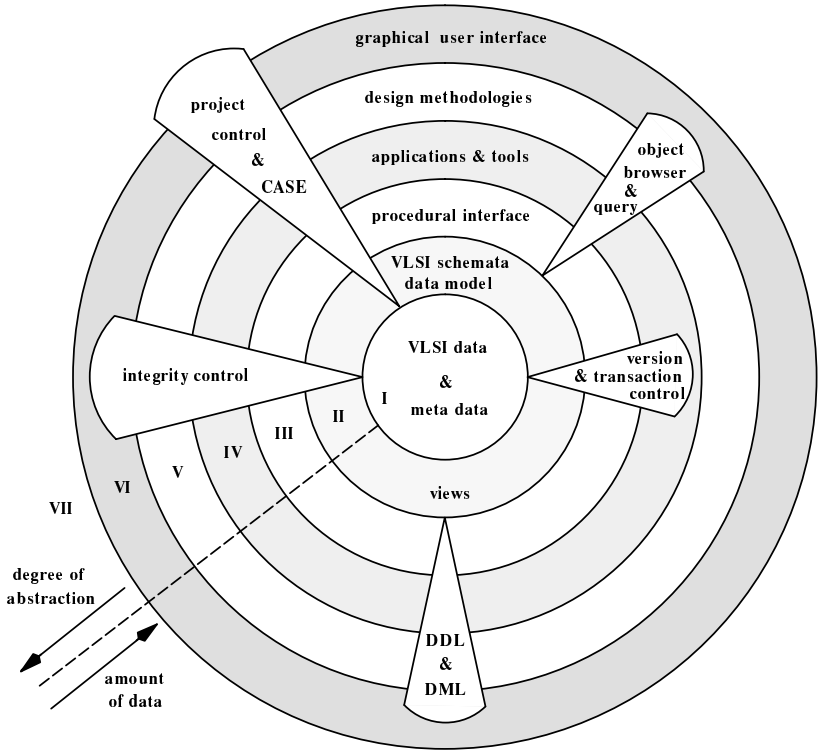

Bild 1: Schalenmodell einer VLSI/CAD-Infrastruktur

VLSI-Entwurfs aus? Welche konzeptionellen Anforderungen an die Modellierbarkeit der VLSI-Daten werden von den Entwurfswerkzeugen und der Datenhaltung gestellt? Die Analyse der Datenstrukturen und Zugriffsmechanismen repräsentativer Entwurfs- und Testwerkzeuge aus dem EDA- und TCAD-Bereich führte auf einen Modellierungsextrakt, der für VLSI-Daten typisch ist.

Generalisierung/Spezialisierung: Die Welt ist vernetzt, auch die technische. Technische Objekte hierarchisch ordnen zu können, setzt ihre Klassifikation voraus. Als mächtiges intellektuelles Modellierungsmittel dient hier die Abstraktion. Global schlägt sich die Generalisierung/Spezialisierung in der Schichtung der VLSI-Abstraktionsebenen nieder: *system*, *algorithmic*, *micro-architectural*, *logic* und *circuit levels*; lokal in der hierarchischen Begriffsbildung: beispielsweise im geometrischen Konstrukt *figure* als Oberbegriff von *circle*, *rectangle*, und *polygon*.

Wesentlich für eine Klassenhierarchie ist ihr semantischer Zusammenhalt, d. h. die *kind-of*-Beziehungen der Klassen untereinander. Sie erlauben die Vererbung von Klasseneigenschaften, was methodisch als Strukturierungsmittel zur Komplexitätsbewältigung und implementierungstechnisch als *code sharing* in Programmen und Datenbanken genutzt wird. Oberklassen stehen für generalisierte Abstraktionen. Unterklassen stellen Spezialisierungen ihrer Oberklassen dar. Verhaltens- und Strukturmuster der Oberklassen werden — eventuell modifiziert — übernommen und eigene hinzugefügt. Generalisierung/Spezialisierung gestattet somit ein Höchstmaß an Ausdrucksökonomie.

Objektausprägung: Der Übergang von der systemanalytischen Phase — der Klassenfindung — zum Entwurf und zur Implementierung geschieht über die *Instanziierung* der Klassen, d. h. über die Objektausprägung oder Exemplarbildung. Dieser Teilungsmechanismus (*instance-of*-Beziehungen zwischen Objekt- und Klassenwelt) unterstützt die Software-Wiederverwendung entscheidend und ist für zellenbasierte Entwurfstechniken (Standardzellen- und Gate-Array-Kundenentwürfe) charakteristisch.

Aggregation: Innerhalb der Objektwelt bedarf es eines Konstruktionsmechanismus für komplexe Objekt-Verbände: *part-of*-Beziehungen ermöglichen zusammengesetzte und geschachtelte Objekte gleicher oder unterschiedlicher Klassenzugehörigkeit. So aggregiert eine *Netzliste* die funktionalen Schaltungskomponenten zu einem

Projekt/Gremium	*Schnittstellen*	*Projektziel/angestrebter Standard*
CFI	II–VII	Industrie-Richtlinien für CAD-Schnittstellen
DoD/EIS	II–VII	dito plus empirischer Nachweis ihrer Brauchbarkeit
ISO/STEP	II–III	Generalstandard für mechanische und elektronische Produktdaten: STEP — Standard for the Exchange of Product Model Data
IEEE/DASS/WGDM	II–VII	Empfehlungen für den Einsatz von Standards in CAD-Infrastrukturen
BMFT/DASSY	I–IV	Modellierung und Standardisierung von Austauschformaten und prozeduralen Werkzeugschnittstellen
EIA/EDIF	III–IV	Standard für den Austausch elektronischer Daten
ESPRIT/ECIP	III–IV	Entwicklung eines formalen konzeptionellen Referenzmodells für alle Arten elektronischer Produkte
ESPRIT/EVEREST	III–IV	Modellierung und Standardisierung von Testdaten

Tabelle 1: Framework-Projekte und Standardisierungsgremien

Objekt-Verband, der als Komplex verwaltet wird. Mit der Art der Aggregation eng verbunden ist die auf ihr definierte operative Semantik, zum Beispiel die Frage, ob aggregierte Objekte (Unterobjekte) beim Löschen des Aggregats (Oberobjekts) mit gelöscht werden.

Abstrakte Datentypen: Diese Anforderung geht zurück auf die Beschränkung konventioneller Datenhaltungssysteme auf wenige Standard-Datentypen, wie Ganz-, Festkommazahlen und Zeichenketten. Abstrakte Datentypen erhöhen die Integrität des Datenbestands, da Operationen und Daten aneinander gebunden sind. Die Implementierung der Datenstrukturen und der zugehörigen Operationen bleibt dem Anwender verborgen: *information hiding*. Beispiel: *Punktlisten* zur Polygon-Beschreibung, wie sie von Layout-Editoren im allgemeinen verwendet werden, mit Operationen, die eine Selbstüberschneidung des Polygons überprüfen und Koordinaten-Transformationen erlauben.

Sichten: Entwurfswerkzeuge arbeiten nur auf bestimmten Repräsentationen eines Entwurfsobjekts: Schaltkreis-Extraktoren benötigen eine andere Objektbeschreibung (Maskengeometrien und Entwurfsregeln) als Logik-Simulatoren (Netzlisten und Verhaltensbeschreibungen). Man unterscheidet zwischen Verhaltens-, Struktur- und physikalischer Domäne. Das Datenaustauschformat EDIF [2] kennt zehn verschiedene *viewTypes*, die hauptsächlich die Entwurfsbereiche Konnektivität, Geometrie und Dokumentation abdecken.

Objekt-Versionierung: Das Archivieren der Entwurfsentscheidungen und der aus ihnen resultierenden Entwurfsobjekte erfordert einen Mechanismus, der die zeitlichen und alternativen Folgen in der Objektentwicklung verwaltet: Versionen und Alternativen. Die Versionsverfolgung ist von zentraler Bedeutung bei der Koordinierung eines Team-Projekts. Damit eng verknüpft sind Entwurfstransaktionen, ein Erfordernis der Datenhaltung, um die Datenkonsistenz bei konkurrierendem Zugriff und Systemfehlern zu sichern.

Interaktionsmechanismen: Konventionelle Techniken des Informationsaustauschs, wie der *Event/Trigger*-Mechanismus in Datenbanken, setzen Kommunikationsmittel und -pfade zwischen den Entwurfsobjekten voraus. Die Datenintegrität zu sichern, ist hier das vorrangige Ziel. Die Korrektheit der Daten muß unter den jeweiligen Aspekten von Aggregation, Typisierung, Sichten, Versionen und Transaktionen gewährleistet sein.

Will man die skizzierten VLSI-Spezifika modellieren, so sind grundsätzlich zwei funktionale Komponenten einer CAD-Infrastruktur zu unterscheiden: Werkzeug- und Datenaustausch-Schnittstellen. Erstere sind prozedural und im Zugriff online. Von der Konzeption her entsprechen sie abstrakten Datentypen: Das Werkzeug kennt nur die Operatoren zum Abfragen und Manipulieren des Datenbestands. Wie der Datenzugriff organisiert ist, bleibt dem Werkzeug verborgen. Datenaustausch-Schnittstellen dagegen sind deskriptiv und arbeiten in der Regel offline. Der Datentransfer läuft geschlossen in einer formatierten ASCII-Datei. EDIF und VHDL sind hier die Standardformate. Beide Aspekte einer CAD-Schnittstelle sind Forschungsgegenstand des DASSY-Projekts.

Die Informationsmodellierung dämmt die Syntax-Inflation ein

Im Bestreben, Form und Inhalt, also Syntax und Semantik, nicht zu vermischen, nimmt das Austauschformat für elektronische Entwurfsdaten, EDIF, eine Vorreiterrolle ein. Um die von vielen

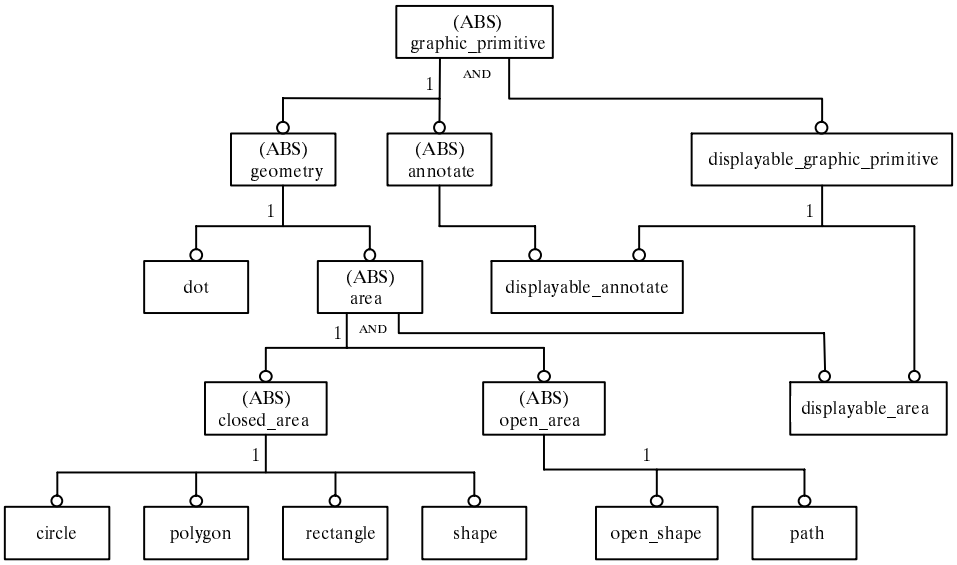

Bild 2: DASSY-Beispiel zur EDIF-Informationsmodellierung

Anwendern prognostizierte Syntax-Krise noch abzuwenden, hat das „EDIF Technical Committee“ bereits 1989 entschieden, daß zukünftige Erweiterungen des Standards nur noch auf der Grundlage eines wohlfundierten Informationsmodells zugelassen werden. Als formales Beschreibungsmittel wurde *Express* [6], eine objektorientierte Modellierungssprache, ausgewählt.

Die „EDIF European TSC Information Modelling Working Group“ (EDIF-IMG) wurde auf Betreiben der DASSY-Partner ins Leben gerufen. Die internationale Arbeitsgruppe befaßt sich ausschließlich mit der Informationsmodellierung der EDIF-Domäne. Die Domäne elektronischer Entwurfsdaten wurde ansatzweise auch in anderen Gremien und Projekten modelliert: *ECAD Domain Model* (EIS-Projekt), *Conceptual Model* (ECIP) und *Test Model* (EVEREST).

Als Ergebnisse der IMG kann festgehalten werden: Informationsmodelle erlauben die syntaxunabhängige Beschreibung der durch das Austauschformat bezeichneten Objekte. Sie formalisieren die Zusammenhänge und Restriktionen in der Objektwelt, spielen beispielsweise eine Referenzrolle, um den Inhalt formatierter Datenbeschreibungen zu vergleichen, und dienen als Grundlage der Syntax-Definition.

Das Express-Modell in Bild 2 zeigt ausschnittsweise Arbeitsergebnisse der EDIF-IMG. Es steht als Beispiel für das *Reverse Engineering* syntaktischer Konstrukte. Hier wurde versucht, das Dickicht der grafischen EDIF-Primitive zu lichten, um eine übersichtliche Darstellung zu erreichen. Die EDIF-Informationsmodelle wurden in [4] dokumentiert.

DaDaMo erfaßt die prozeduralen Aspekte in einem CAD-Framework

Entwurfswerkzeuge sollen über die Framework-Schnittstellen auf einem gemeinsamen Datenbestand operieren und gegebenenfalls austauschbar sein. Prozedurale Werkzeugschnittstellen leisten das Gewünschte. Bild 3 zeigt den Aufbau einer solchen, wie sie im DASSY-Projekt realisiert wurde [5].

Die Aufgabenstellung ähnelt der geschilderten Informationsmodellierung für Daten-Austauschformate, nur daß hier der dynamische Aspekt des Datenzugriffs hinzukommt. Das DASSY-Datenmodell, DaDaMo, ist ein effizienter Konzept-Mix aus der semantischen und objektorientierten Datenmodellierung. Seine Implementierbarkeit wurde bereits prototypisch nachgewiesen und seine Relevanz für die Framework-Standardisierung international anerkannt. DaDaMo beinhaltet erweiterte Konzepte für die Definition und Manipulation komplexer Entwurfsdaten.

Entwurfsdaten, wie Schaltungskomponenten und Verbindungsstrukturen, werden grundsätzlich als *Objekte* im Sinne des Objekt-Paradigmas modelliert (Datenkapselung, Klassenzugehörigkeit) und gehandhabt (abstrakte Datentypen, klassenbezogene Methoden). Die strukturellen Eigenschaften eines Objekts werden durch seine At-

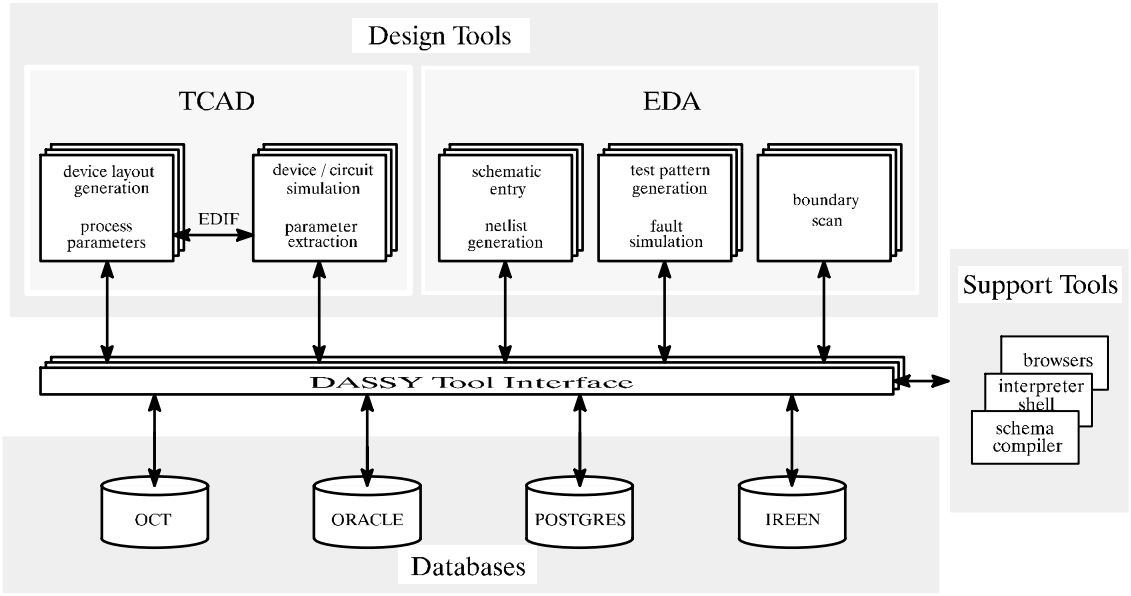

Bild 3: DASSY-Prototyp: Szenario

tribute und attribuierbaren Beziehungen zu anderen Objekten beschrieben, die dynamischen durch benutzer-definierte Operationen. Die Definition von *Constraints* erlaubt darüber hinaus, die Objekteigenschaften individuell einzuschränken, um den besonderen Anforderungen der Entwurfswerkzeuge zu genügen. Der Konsistenzmechanismus in DaDaMo basiert auf dem Constraint-Konzept. Die Constraints werden ereignisabhängig ausgelöst und durch die Ausführung benutzerdefinierter Methoden evaluiert.

DaDaMo unterstützt die Modellierung von VLSI-spezifischen Weltausschnitten, indem es die multiple Vererbung der Objekteigenschaften zuläßt und die Beziehungstypen frei definierbar hält. Welche Semantik eine Beziehung trägt und welche Objektstrukturen darstellbar sind, bestimmt der Anwender. Die operative Semantik der Beziehungen wird aktiv unterstützt. Im Gegensatz zu anderen Vorschlägen drückt DaDaMo Beziehungen nicht allein durch die Angabe von Objektreferenzen aus, verbirgt das, was auf den Beziehungen geschehen soll, nicht in unzugänglichem Operationscode oder beläßt es gar auf bloße Hinweise durch entsprechende Bezeichner. Beziehungstypen, die oft im Mikroelektronik-Entwurf benötigt werden, sind in DaDaMo bereits vordefiniert:

relate: symmetrischer Beziehungstyp (nur ein Rollenname für beide Richtungen), ohne besondere operative Semantik, dient der Navigation auf assoziierten Objektmengen durch Operationen wie *set_relation, get_relation, modify_relation* und *remove_relation*;

contain: abgeleitet vom Typ *relate*, beschreibt die Beziehung zwischen einem übergeordneten Container-Objekt, dem Aggregat, und den aggregierten inneren Objekten, wobei der Zugriff auf das Container-Objekt auch den Zugriff auf die inneren Objekte und deren Beziehungen impliziert (durch Überschreiben von *get_relation*);

shared contain: abgeleitet vom Typ *contain*, wobei aber die Existenz der inneren Objekte vom gesamten Aggregat abhängt: wird das Container-Objekt gelöscht, so auch seine Unterobjekte (durch Überschreiben von *remove_relation*), es sei denn, sie sind gleichzeitig innere Objekte anderer Container-Objekte;

strong contain: abgeleitet vom Typ *shared contain*, wobei ein inneres Objekt durch spezielle Klassen-Constraints ausschließlich nur einem Container-Objekt zugeordnet ist und somit grundsätzlich mit diesem gemeinsam gelöscht wird.

Aus den vorgegebenen Beziehungstypen können weitere abgeleitet und durch benutzereigene Methoden in ihrer operativen Semantik differenziert werden.

Berechnete Attribute verringern die Datenredundanz und erhöhen die Datenkonsistenz. Konkrete Entwurfsentscheidungen können durch Attribute, berechnet auf der Objektebene, d. h.

wertmäßig zur Laufzeit der Applikation, unmittelbar weitergegeben werden. Dies vereinfacht das Management für eine Objektversionierung erheblich. Die für die Ableitung eines Attributwerts von dem eines anderen Objekts erforderlichen Berechnungsformeln werden im Schema angegeben. VLSI-Beispiel: Berechnung der kapazitiven Belastung eines Schaltungsknotens durch die Anzahl der aktuell verdrahteten Zellanschlüsse.

DaDaMo bietet nach außen zwei Sprachschalen für die Schemabeschreibung an. Zum einen steht eine Express-orientierte Schale zur Verfügung, um die in Express geschriebenen Informationsmodelle nahtlos in die Werkzeugschnittstelle zu integrieren, und zum andern eine an C++ angelehnte Sprachschale, da der Großteil aller industriellen Framework-Applikationen in C bzw. C++ geschrieben ist.

Als die Modelle laufen lernten ...

Wir sind unisono mit David Harels Antwort auf Brooks' Pessimismus: *Biting the Silver Bullet: Toward a Brighter Future for System Development* [3]: Der „Akzidensteil" der Software-Entwicklung, die zufälligen Begleiterscheinungen und Schwierigkeiten im Umgang mit komplexen Sachverhalten, ist beherrschbar oder wird es in naher Zukunft sein. Grafik-Tools, wie Generatoren, Editoren und Browser, und Management-Tools für die Versionsverfolgung und Konfiguration von Objektenmengen erleichtern dem Weltmodellierer die Arbeit. Konstruktion, Dokumentation und formale Analyse seiner Modelle werden bereits auf dem Rechner durchgeführt. Was stets schwierig bleiben wird, ist der „Essenzteil", die in Struktur und Verhalten liegende Komplexität des Weltausschnitts.

Was fehlt sind Code erzeugende Werkzeuge, die das konzeptionelle Modell „zum Laufen bringen", d. h. in die Konstrukte konventioneller Hochsprachen bzw. für VLSI-Applikationen in eine Hardware-Beschreibungssprache compilieren. Das sind zum Beispiel Schema-Compiler, die ein Weltmodell automatisch in ein Datenbankschema überführen, oder Modell-Simulatoren, die eine Aussage über die Konsistenz der gespeicherten Objekte und der Interaktionen auf ihnen erlauben. Das heißt also: Rechnerunterstützung nicht nur bei der textuellen und grafischen Darstellung von Weltmodellen, sondern auch bei der Inspektion und Implementierung dieser.

Literatur

[1] Brooks, Frederick P.: *No Silver Bullet: Essence and Accidents of Software Engineering.* Computer, Jg. 20, H. 4, 1987, S. 10–19

[2] *Electronic Design Interchange Format EDIF, Version 2 0 0, EIA/ANSI Standard 548.* Electronic Industries Association, Washington D. C., 1990

[3] Harel, David: *Biting the Silver Bullet: Toward a Brighter Future for System Development.* Computer, Jg. 25, H. 1, 1992, S. 8–20

[4] *IMG Information Model of EDIF Version 2 0 0, Version 1.0.* Hrsg.: Wilkes, Wolfgang, FernUniversität Hagen, 1992

[5] Quibeldey-Cirkel, Klaus: *Verbundprojekt DASSY: Datentransfer und Schnittstellen für offene integrierte VLSI-Entwurfssysteme.* BMFT-Forschungsbericht, TIB Hannover, 1993

[6] Schenck, Douglas: *EXPRESS Language Reference Manual.* McDonnell Aircraft Company, St. Louis, MO., 1990