MIDGET - Ein flexibles, simulationstechnisches Entwicklungssystem

Magnus Rimvall, Francois Cellier
Institut für Automatik und Industrielle Elektronik
Eidgenössische Technische Hochschule (ETH)
CH-8092 Zürich, Schweiz
Tel. 01/256 28 42

1. Einleitung

In den letzten zehn Jahren hat eine rasche Entwicklung auf dem Gebiet der Simulationssoftware stattgefunden. Parallel zum Aufschwung von strukturierten allgemeinen Sprachen wie PASCAL und ADA sind flexiblere und besser strukturierte Simulationssprachen auf der Markt gekommen. In vielen dieser Sprachen kann der Benützer interaktiv arbeiten. Er kann, ausgehend von graphischen Resultaten einer Simulation, direkt Parameterwerte ändern und neue Simulationsläufe ausführen lassen. Die Modellbeschreibung kann jedoch normalerweise nicht interaktiv eingegeben oder geändert werden, sondern muss off-line, unter Verwendung eines Texteditors erstellt werden. Ferner müssen meistens andere Vorgänge, wie Kompilieren und Linken, ausgeführt werden, bevor man mit der Simulation beginnen kann. Für den Benützer bedeutet dies, dass er nicht nur die verwendete Simulationssprache beherrschen muss, sondern dass er auch mit dem Betriebsystem ('wie erzeuge ich ein Datenfile?') und der Rechnerkonfiguration ('wo finde ich die Plotbibliothek? wie heisst der Printer an der Satellitenstation?') vertraut sein muss. Aber, da es den meisten Benützern von Simulationssoftware schliesslich egal ist, wie und auf welchem Rechner er zu seinen Resultaten kommt, könnte sehr viel Zeit gespart werden, wenn die Umgebung einfach und transparent wäre. In diesem Vortrag werden eine solche Umgebung, das Entwicklungssystem MIDGET, präsentiert und einige allgemeine Aspekte der Simulationssprachumgebung diskutiert.

2. Merkmale eines Entwicklungssystems für Simulationstechnik

Moderne Simulationssprachen wie etwa ACSL und CSSL-IV bieten dem Benützer während der Ausführung von Simulationen einen sehr hohen interaktiven Komfort. In der Einleitung mussten wir aber feststellen, dass sowohl vor als auch nach der eigentlichen Simulation einige Hürden, die nichts mit Simulationstechnik zu tun haben, zu überwinden sind. Unter Verwendung eines simulationstechnischen Entwicklungssystems könnte man aber diese Hürden vermeiden. Ein solches System sollte die folgenden Anforderungen erfüllen:

- Oft muss ein Benützer das gleiche Simulationspaket auf verschiedenen Rechneranlagen mit unterschiedlichen Peripheriegeräten (Terminals und/oder Plotters) benützen. Um für den Benützer dieses Umsteigen zwischen den verschiedenen Rechnern zu erleichtern, muss das Entwicklungssystem auf allen Rechnern gleich aussehen (gleiche Benützerschnittstelle).

- Die Entwicklungssysteme für verschiedene Simulationssprachen sollen ähnlich gestaltet sein, so dass der Benützer schnell zwischen diesen Sprachen wechseln kann.

- Das Entwicklungssystem muss alle Befehle anbieten, die der Benützer während der Vorbereitung, dem Ausführen und der Analyse der Simulation braucht.

- Um das Erlernen zu erleichtern, soll jedes Entwicklungssystem so einfach wie möglich gestaltet sein; interaktive HELP-Dokumentation soll jederzeit zur Verfügung stehen, und möglichst viele Eingabefehler sollen schon vom Entwicklungssystem abgefangen werden.

Da die beiden letzten Anforderungen zum Teil widersprüchlich sind, muss bei der Herstellung jedes Entwicklungssystems gut überlegt werden, welche Möglichkeiten der Benützer unbedingt braucht. Unsere Erfahrung zeigt aber, dass für unsere simulationstechnischen Entwicklungssysteme (im Unterschied zu allgemeineren Verwaltungssystemen für grössere Softwareprojekte) eine relativ einfache Benützerschnittstelle in Form eines einzigen Menus völlig ausreichend ist. Im nächsten Abschnitt wird ein solches System, MIDGET, präsentiert.

3. MIDGET

MIDGET (Menu-driven Interactive Development-System for Generic Engineering Tasks) ist ein Entwicklungssystem, das besonders für Simulationsanwendungen konzipiert worden ist. MIDGET bietet dem Benützer eine einfache, schnell erlernbare Schnittstelle zu der verwendeten (Simulations-)Software, was besonders für den Anfänger und den Gelegenheitsbenützer von Nutzen ist.

Anhand eines Beispiels wollen wir jetzt MIDGET kurz vorstellen. Nehmen wir an, dass wir unter Verwendung von MIDGET ein Simulationsmodell aufstellen möchten, um einige Simulationen durchzuführen. Um Zugriff zu den MIDGET Entwicklungssystemen zu bekommen, müssen wir dann zuerst den Befehl DEV (DEVelopment Systems) eingeben. MIDGET antwortet mit dem folgenden Menu (die Benützereingaben sind unterstrichen):

```
$ DEV
SELECT DEVELOPMENT SYSTEM : (A)  ACSL
                            (G)  GASP
                            (R)  SDL
                            (S)  SLAM
                            (T)  TEX
                            (X)  SYNTAX
                            (.)  USER DEFINED
                            (EXIT)  EXIT
--------------------------->: A
```

Wir wählen A für ACSL, MIDGET initialisiert das ACSL-Entwicklungssystem, begrüsst uns und listet allfällige Änderungen auf:

```
************************************************************
*****                                                  *****
*****      WELCOME TO THE ACSL STUDENTS ACCOUNT        *****
*****                                                  *****
************************************************************

Message of the day:                 8. Juni 1984
*******************

  1)  Sie arbeiten jetzt mit der Version 8E/8F von ACSL.
  2)  Fehler und weitere Wuensche bitte an uns rapportieren.
                            M. Rimvall      Tel.: 28'42
                            F. Cellier      Tel.: 42'81
```

Jetzt sind wir im eigentlichen Entwicklungssystem und MIDGET meldet sich mit dem ACSL-Hauptmenu (aus typographischen Gründen geringfügig geändert verglichen mit dem interaktiven Menu):

```
PRESENT SYSPICS :
 CURRENT ACSL PROBLEM:  ***UNDEFINED
 ACSL LISTING CURRENTLY:  ON
```

```
POSSIBLE ACTION :
 (A) RUN ACSL PROBLEM                    (R) READ FILE FROM OTHER PROBLEM
 (D) DELETE ACSL PROBLEM                 (S) SELECT ACSL PROBLEM
 (E) EDIT ACSL PROBLEM FILE              (T) DISPLAY STATUS OF QUEUES
 (F) EDIT ACSL DATA FILE                 (V) DISPLAY VAX-SPECIFIC INFO.
 (G) DISPLAY GENERAL HELP INFORMATION    (W) WRITE FILE TO OTHER PROBLEM
 (I) SWITCH ACSL LISTING GENERATION      (X) EXECUTE OLD PROBLEM ONCE MORE
 (J) EDIT ACSL LISTING FILE              (Z) EDIT THE LINK FILE
 (L) LIST OF EXISTING ACSL PROBLEMS      (SP)  SPAWN NEW PROCESS
 (M) DISPLAY THE MESSAGE OF THE DAY      (MENU) SWITCH MENU ON/OFF
 (N) PRINT NON-ACSL FILE (AFTER ERROR)   (DEV) CHANGE DEV. SYSTEM
 (O) MAKE OLD VERSION CURRENT AGAIN      (EXIT) EXIT FROM DEV. SYST.
 (P) PURGE OLD VERSIONS                  (HELP) HELP ON THESE COMMANDS
 (Q) SHOW DISK QUOTA
------------------------------------>:
```

Ausgehend von diesem Menu können wir jetzt jeden Befehl, den wir bei der Modellformulierung/Simulation brauchen, ausführen lassen. Nach dem Ausführen des Befehls kommen wir automatisch zu diesem Menu zurück. Für den erfahrenen Benützer besteht die Möglichkeit, durch den Befehl MENU das Auflisten des Menus zwischen jeder Befehlseingabe zu unterdrücken; MIDGET meldet sich dann nur mit:

```
PRESENT SYSPICS :
 CURRENT ACSL PROBLEM:  ***UNDEFINED
 ACSL LISTING CURRENTLY:  ON

PRESS <CR> FOR MENU-------->:
```

Das Menu kann jederzeit mit dem gleichen Befehl MENU wieder eingeschaltet werden; für ein einziges Auflisten des Menus reicht aber ein einfacher Tastendruck der RETURN-Taste.

Sämtliche MIDGET Entwicklungssysteme arbeiten mit einem oder mehreren sogenannten SYSPIC's, Zeiger zu dem gerade behandelten Problem. Dies bedeutet, dass wir nur einmal (zu Anfang jeder Session) mit dem S (SELECT) Befehl ein Problem (Programm) auswählen müssen. Danach übernimmt MIDGET die Verantwortung und garantiert, dass immer die richtigen Files editiert, übersetzt, ausgeführt oder ausgedruckt werden. Ferner wird durch MIDGET das Auseinanderhalten von verschiedenen Filetypen gewährleistet. Der Benützer muss nicht mehr im Kopf behalten, ob jetzt das gewünschte Listing im File REAKTOR.LIS oder REAKTOR.LST zu finden ist.

In unserem Fall möchten wir mit dem S-Befehl das Simulationsmodell PILOT auswählen (hätten wir die Übersicht über die existierenden Probleme verloren, könnten wir diese mit dem L Befehl auflisten):

```
PRESENT SYSPICS :
 CURRENT ACSL PROBLEM:  ***UNDEFINED
 ACSL LISTING CURRENTLY:  ON

PRESS <CR> FOR MENU-------->: S
Enter name of ACSL problem (without extension) : PILOT

PRESENT SYSPICS :
 CURRENT ACSL PROBLEM:  PILOT
 ACSL LISTING CURRENTLY:  ON

PRESS <CR> FOR MENU-------->: A
```

MIDGET markiert unsere Wahl von SYSPIC. Wenn dies ein neues Programm wäre, müssten wir jetzt mit dem E Befehl einen geeigneten Editor aufrufen und unsere ACSL Systembeschreibung eingeben (MIDGET wählt je nach verwendetem Terminaltyp automatisch einen geeigneten Editor mit dem Terminal angepassten Editparametern aus). Mit dem Befehl A können wir jetzt den Befehlsablauf, der uns zum interaktiven Modus von

ACSL führt, starten. Im Normalfall beinhaltet dieser Ablauf mehrere, von Simulation zu Simulation verschiedene Kompilationen und Link-Befehle. MIDGET trifft aber immer die richtige Entscheidung darüber, welche Befehle ausgeführt werden müssen.

Nach der Simulation kommen wir wieder zu MIDGET zurück. Hier werden uns zwei Fragen gestellt; die erste Frage erlaubt uns, die Resultate (numerisch und graphisch) ausgedruckt auf Papier zu erhalten, und die zweite gibt uns eine aktive Aufmunterung zum Speicherplatzsparen:

```
END ACSL RUN
Do you wish to receive a hardcopy? (Y/N): N
Do you wish to clean up your files? (Y/N): Y
```

Die weiteren Befehlen des Menus können in vier Gruppen aufgeteilt werden:

1/ Befehle, die uns helfen, Fehler in unseren Programmen zu finden,
2/ Befehle, die uns erlauben, ACSL besser auszunützen (fremde Programme dazuzulinken u.s.w.)
3/ Befehle, die verschiedene Statusinformationen zurückgeben, und
4/ Systembefehle, die uns erlauben, MIDGET zu verlassen.

Dank der zweiten Gruppe können nicht nur Anfänger, sondern auch erfahrene Simulationsspezialisten bei der Verwendung von MIDGET profitieren. Volles Ausnützen eines grossen Systems wie ACSL setzt normalerweise auch gute Betriebssystemkenntnisse voraus; der Benützer gewinnt aber Zeit, wenn er sich diese Kenntnisse nicht etwa selber zulegen muss, sondern sich auf die Systemkenntnisse des MIDGET-Konstrukteurs verlässt.

Ein weiterer Vorteil von MIDGET ist, dass sich die Entwicklungssysteme der verschiedenen Simulationssprachen sehr ähneln, was das Umsteigen zwischen verschiedenen Simulationspaketen erheblich vereinfacht. Wenn man z.B. die Entwicklungssysteme von ACSL und GASP vergleicht, merkt man, dass 20 von den 25 Befehlen aus der Sicht des Benützers identisch sind.

Bis jetzt stehen MIDGET Entwicklungssysteme für die Simulationssprachen ACSL, GASP und SLAM und für das Simulationsdatenbankpaket SDL zur Verfügung. Ferner wird die Implementation weiterer Entwicklungssysteme dank des Vorhandenseins eines "Entwicklungssystems-Entwicklungssystems" stark vereinfacht. Dieses System stellt ein Rahmenprogramm zur Steuerung des neuen Entwicklungssystems zur Verfügung, hilft dem Benützer bei der Überprüfung der Korrektheit der Befehlsabläufe und bietet Unterstützung bei der Übertragung des Entwicklungssystems auf andere Rechneranlagen.

4. MIDGET auf einem Mehrbenützersystem

Inbesonders an den Hochschulen müssen, meistens aus administrativen Gründen, mehrere Benützer (Studenten) die gleiche Kontonummer verwenden, was zu einer Reduktion der Datensicherheit führt. MIDGET unterstützt eine aktive Datensicherung, indem jedem Benützer eine sogenannte "Subkontonummer" und ein eigener Speicherbereich zugewiesen wird. Er hat nur zu den eigenen Files Zugriff; ausserdem hat er keine Möglichkeit, MIDGET zu verlassen, ohne die Session zu beenden. Das Einrichten neuer Subkontonummern wird unter MIDGET zu einer Trivialität. Durch das Löschen alter Subkontonummern verhindert man ferner den Missbrauch von früheren, nicht mehr berechtigten Benützern.

MIDGET enthält auch andere Hilfsmittel für Mehrbenützeranlagen. Es ist z.B. möglich, die zugänglichen Entwicklungssysteme von Kontonummer zu Kontonummer unterschiedlich zu beschränken und auch innerhalb eines Entwicklungssystems ausgewählte Befehle nur gewissen (priviligierten) Benützer zur Verfügung zu stellen. Priviligierte Benützer haben alle Entwicklungssystemen zur Verfügung und können auch MIDGET verlassen, ohne dass sie vom Rechner "ausgelogt" werden.

5. MIDGET beseitigt Systemabhängigkeiten

Die meisten modernen Simulationssprachen beinhalten graphische Ausgabemöglichkeiten. Leider gibt es auf Grund der schnellen technischen Entwicklung und einer fehlenden Koordination eine Vielzahl von verschiedenen graphischen Normen. Aus diesem Grund muss jedes kommerzielle Simulationspaket mehrere graphische Schnittstellen unterstützen. Das Paket muss ferner bei jeder Simulation vom Benützer Information erhalten, welche Plotnorm verwendet werden soll. Ähnlich brauchen die meisten Editoren, inbesondere die flexibleren full-screen Editoren, Auskunft über den verwendeten Terminaltyp. Da die meisten grösseren Rechenanlagen eine Vielfalt von Terminaltypen besitzen, muss, je nach Terminaltyp, der Benützer verschiedene Editoren aufrufen, verschiedene Plot-Pakete zum Programm linken u.s.w. MIDGET beseitigt diesen Dschungel von gerätabhängigen Befehlen; der Plottertyp wird bei der MIDGET Installation festgelegt und den Terminaltyp muss man nur einmal (beim Login) angeben.

6. Rechnerportabilität von MIDGET

MIDGET bietet nicht nur dem Benützer von Simulationssystemen und dem Konstrukteur neuer Entwicklungssysteme viel Komfort, sondern auch der "System-Manager", der MIDGET auf einen neuen Rechner installieren muss, bekommt von MIDGET eine aktive Unterstützung. Im Unterschied zu den meisten grösseren Software-Systemen, die mit einer dicken Installationsbeschreibung, die bis zum letzten Punkt befolgt werden muss, geliefert werden, ist die Installationsanleitung von MIDGET sehr dünn. Man muss MIDGET nur vom Band lesen, eine "Command-procedure" (Betriebsystemsprogramm) starten und diesem Programm einige Fragen bezüglich der Rechnerkonfiguration beantworten. Danach installiert MIDGET sich selbst, editiert sogar einige eigene Files selber um, so dass diese auf der Zielrechnerkonfiguration verwendbar sind. Da MIDGET bei jedem Login initialisiert wird, wird auch das sog. Login-file automatisch umeditiert (neu kreiert). Wenn die Lizenzbestimmungen dies erlauben, installiert MIDGET sogar die Zielsysteme der mitgelieferten Entwicklungssysteme. Zuletzt wird die Installation getestet, indem sich MIDGET selber startet.

7. Einsatz von MIDGET

Ein Vorläufer von MIDGET, ein Entwicklungssystem für das Simulationspaket ACSL, wurde Ende 1982 an der ETH-Zürich entwickelt. Dieses System war primär für Studenten einer Einführungsvorlesung in Simulationstechnik konzipiert, die auf einer für sie unbekannten Rechneranlage kleinere Simulationsübungen durchführen mussten.

Der erfolgreiche Einsatz dieses MIDGET-ähnlichen Systems zeigte, wie man unerfahrenen Personen einen unmittelbaren Zugriff zu einem Softwarepaket geben kann, ohne dass diese zuerst während etlicher Stunden Betriebssystem und Rechnerkonfiguration studieren müssen. Aus diesem Grund und weil auch erfahrene Simulationstechniker entdeckt hatten, wie komfortabel man mit diesem Entwicklungssystem Zugriff zu ACSL bekommt, wurde beschlossen, ein generelles Entwicklungssystem (MIDGET) zu entwerfen. Seit Mitte 1983 steht MIDGET zur allgemeinen Verfügung an der ETH, ferner ist MIDGET an mehr als einem Dutzend externen Rechnern weltweit installiert worden.

MIDGET wurde auf einer VAX-Anlage implementiert. Da MIDGET sehr viele Systemoperationen ausführen muss (Kompilieren, Linken u.s.w.), ist ein grosser Teil von MIDGET in DCL (VAX/VMS Betriebsystemsprache) geschrieben, aus Portabilitäts- und Geschwindigkeitsgründen ist aber möglichst viel in Pascal implementiert. Obwohl die jetzige Version von MIDGET nur unter VAX/VMS läuft, ist das Paket in jede flexiblere Operativsystemsprache (z.B. UNIX/C, IBM/JCL) übersetzbar.

8. Referenzen

Cellier, F.E; New Problems in Software Complexity; Simulation, Vol 41 No 3; 1983.

Rimvall, M; MIDGET - A User's Guide; Interne Publikation, ETH-Zürich, 1983.