

4.3 Typisierung der Simulationssoftware

- **Entwicklungsphasen der Simulationssoftware:**

- Simulationssoftware mit Entwicklungssprüngen in 10-Jahresabständen (50-ziger Jahre bis 90-ziger Jahre)
- 5 “Generationen” der Simulationssoftware
 - » nach *Prof. B. Schmidt, Uni Passau*

Phase (1): Simulation in höheren Programmiersprachen

- Keine simulationsspezifische Unterstützung
- Vollständige Eigenentwicklung der Simulationsfunktionalität
- Vorteile:
 - » Nutzung vorhandener Programmierkenntnisse in vertrauter Programmiersprache
 - » Keine Beschaffungskosten für spezielle Simulationssoftware
 - » Transparenz aller simulationsspezifischen Funktionen
 - » Universelle Anwendbarkeit

Entwicklungsphasen der Simulationssoftware

Phase (1): Simulation in höheren Programmiersprachen

- Nachteile:
 - » Sehr zeitaufwendige Modellimplementation
 - » Hohe Fehleranfälligkeit der simulationsspezifischen Funktionen
 - Beispiele: Zufallszahlenerzeugung, Ereignislistenverwaltung
 - » Oftmals schwierige Modellmodifikation
 - » Sehr gute Programmierkenntnisse erforderlich
 - » Eingeschränkte Wiederverwendbarkeit der Simulationsprogramme
 - » Geringe Einheitlichkeit der individuellen Simulationsprogramme
- Nutzung verschiedenster Programmiersprachen:
 - » Fortran, Pascal, Modula-2, C++, kürzlich auch Java
 - » Fortran-Simulationsprogramme bis heute im Einsatz (vor allem im technischen Bereich)

Entwicklungsphasen der Simulationssoftware

Phase (2): Simulationspakete / -sprachen

- Bereitstellung simulationsspezifischer Funktionalität
 - » Ablaufsteuerung (Ereignislistenverwaltung)
 - » Unterstützung bestimmter Simulationsweltbilder
 - » Simulationsspezifische Verfahren: Zufallszahlenerzeugung, statistische Berechnungen, Ausgabefunktionen
- Simulationspaket =
Sammlung von Prozeduren/ Methoden und Funktionen für simulationsspezifische Aufgaben
 - » *Simulationsbibliothek*
- Implementierungssprache = Basissprache des Simulationspaketes
 - » Erstellung eines Simulationsprogramms in der Basissprache
- Aufrufe der in der Simulationsbibliothek enthaltenen simulations-spezifischen Prozeduren/ Methoden und Funktionen

Entwicklungsphasen der Simulationssoftware

• Simulationspakete

- Vorteile:
 - » Flexibilität der Basissprache voll erhalten
 - » Universelle Anwendbarkeit durch problemlose Erweiterbarkeit des Simulationspaketes
 - » Aufbau auf vorhandenen Programmierkenntnissen möglich
 - » Geringerer Implementationsaufwand gegenüber höheren Programmiersprachen
- Nachteile:
 - » Gute Programmierkenntnisse und große Simulationserfahrungen erforderlich
 - » Anpassung an vorgegebenes Simulationsweltbild
- **Beispiele:** GPSS-FORTRAN, DEMOS, DESMO (M2, C++, J),....

Entwicklungsphasen der Simulationssoftware

- **Simulationssprachen**

- = Speziell als Implementierungssprachen für Simulationsmodelle konzipierte Programmiersprachen mit simulationsspezifischen Sprachkonstrukten auf Anweisungsebene
- Eigene Sprachcompiler
- Als universelle oder anwendungsbezogene Simulationssprachen
 - » Zusätzlich zu allgemeinen Simulationsfunktionen z.T. modellklassenspezifische Sprachelemente, die Modellerstellung unmittelbar erleichtern
 - » Komplette Modellkomponenten, -funktionen
 - Beispiel: Bedienstationen mit integrierten Warteschlangen
 - » Höherer Grad an Problemorientierung bei verringerter Flexibilität
 - » Spezialsprachen zur Simulation von Rechensystemen, etc.
- Vorteile:
 - » Geringerer Implementierungsaufwand (Faktor 3 bis 9)

Entwicklungsphasen der Simulationssoftware

- **Simulationssprachen**

- Nachteile:
 - » Anpassung an verfertigtes Simulationsweltbild
 - » Beschaffung spezieller Compiler notwendig
 - » Hoher Einarbeitungsaufwand in neue Sprache
 - » Noch größere Programmierkenntnisse erforderlich
- Beispiele: Simscript, Simula, GPSS,

Entwicklungsphasen der Simulationssoftware

Phase (3): Sprachen für kombinierte Systeme

- Kombination diskreter und kontinuierlicher Simulationsansätze für bzw. verschiedene Weltbilder in einer gemeinsamen Software
- Vorteile:
 - » Unterstützung der Modellierung kombinierter Systeme durch eine gemeinsame Software
 - » Realitätsnähere Abbildung kombinierter Systeme
 - » Erleichterung der Behandlung von Diskontinuitäten
- Nachteil: Höherer Einarbeitungsaufwand in komplexere Software
- **Beispiele:** GASP IV, SLAM, SIMAN,...

Entwicklungsphasen der Simulationssoftware

Phase (4): Anwendungsspezifische Simulatoren

- Simulationssoftware für spezielle Anwendungsfelder
 - » Vorrangig für Fertigungs-, Regelungstechnik, Rechensysteme
 - » Zusätzlich Funktionen zur **Animation**
Bessere Nachvollbarkeit des Ablaufgeschehens von Modellen direkt am Bildschirm
- Vorteile:
 - » Wesentlich schnellere und komfortablere Modellerstellung
 - » Benutzer arbeitet mit ihm verständlichen. problemspezifischen Bausteinen
 - » Geringere fachliche Anforderungen in Simulationstechnik
- Nachteile:
 - » beschränkte Flexibilität und Abbildungsgenauigkeit
 - » häufig lange Rechenzeiten
 - » z.T. hohe Softwarekosten
- **Beispiele:** SEE WHY, SIMFACTORY, XCELL, NETWORK,...

Entwicklungsphasen der Simulationssoftware

Phase (5): Integrierte Simulationssysteme

- Softwareunterstützung des gesamten Modellbildungszyklus
 - » Datenaufbereitung und Datenmanagement,
 - » Modellspezifikation,
 - » Hierarchische Modellierung,
 - » Experimentierumgebung,
 - » Statistische Ergebnisauswertung,
 - » Grafikunterstützung:
 - » Animation,
 - + grafische Modellierung,
 - + grafische Benutzungsoberflächen
 - » Teilweise Objektorientierung
- Offene Systeme: Schnittstellen zu
 - » Datenbanken, Tabellenkalkulation,
 - » Grafik-, Statistiksystemen

Entwicklungsphasen der Simulationssoftware

Phase (5): Integrierte Simulationssysteme, Forts.

- Vorteile:
 - » Erhöhung der Leistungsfähigkeit und Bedienungsfreundlichkeit
 - » Umfassende Softwareunterstützung eines Simulationsprojektes in einer einheitlichen Softwareumgebung
 - » Konzentration des Benutzers auf Abbildung der relevanten Systemkomponente, nicht auf die eigene Programmierung
 - » Offene Systeme mit definierten Schnittstellen für Erweiterungen
- Nachteile:
 - » Hoher Lernaufwand umfassendes, komplexes Softwaresystem
 - » Oftmals hohe Kosten für Beschaffung und Betrieb
 - » Gefahr der Erzeugung falscher Ergebnisse bei mangelnder Erfahrung
- **Beispiele:** CREATE!, SIMPLE++, SIMPLEX II,...

Entwicklungsphasen der Simulationssoftware

Zwei signifikante Entwicklungsrichtungen heutiger Simulationssysteme

- **Universelle Simulationswerkzeuge**
 - Entwickelt aus höheren Programmiersprachen bzw. Simulationssprachen für allgemeine Simulationszwecke
 - Große Anwendungsbreite
 - Hohe Anforderungen an die Programmierkenntnisse der Benutzer
- **Anwendungsorientierte Simulationswerkzeuge**
 - Aufbauend auf Programmier-/ Simulationssprachen für spezielle Anwendungsfelder
 - Hohe Anwendungsnähe und Modellierungseffizienz
 - Beschränkte Flexibilität und Anwendungsbreite

Entwicklungsphasen der Simulationssoftware

- **Generationen der Simulationssoftware nicht nur historisch, sondern auch heute noch interessant**
 - Simulationssysteme mit eigenen Modellierungssprachen bzw. mit Schnittstellen zu höheren Programmiersprachen für Systemerweiterungen
 - » Simulationsprogrammierung notwendig
 - Heutige Simulationswerkzeuge häufig mit eigenen Entwicklungsgenerationen
 - » Entstehung aus klassischen Simulationssprachen
 - » Heute mit typischen Funktionen der letzten Softwaregeneration:
 - + Integrierte Programmierumgebung
 - + Grafik, Animation, Grafische Benutzungsoberflächen,...
 - + Hierarchische Modellstruktur, etc.
 - » Beispiele: GPSS (-> GPSS PC), SLAM (-> SLAMSYSTEM), SIMAN (SIMAN V, ARENA),...

Entwicklungsphasen der Simulationssoftware

- **Obige Gliederung der Simulationssoftware nach Generationen eher idealtypisch**
 - Bei modernen Systemen verschmelzen Softwaregenerationen
 - Viele Simulatoren nachträglich erweitert um eigene Modellierungssprachen
 - » **Beispiel: SIMFACTORY II.5 Rel. 6**
 - Alternativ: Schnittstellen zu höheren Programmier- bzw. Simulationssprachen
 - » **Beispiel: SIMAN V**
 - Kombination benutzereigener Modellkomponenten mit Standardbausteinen der anwendungsorientierten Simulationssysteme möglich
 - » **Größere Flexibilität durch Erweiterbarkeit**
 - » **Ausweitung des Anwendungsspektrums**
 - Objektorientierte universelle Simulationssysteme mit vordefinierten Grundbausteinen, die für spezielle Anwendungen objektorientiert erweitert werden können

Zum Stellenwert der Animation

- **Leistungsfähige Grafik wichtiger Faktor für wachsende Bedeutung der Simulationstechnik**
- **Animation**
 - = **Grafische Darstellung der Zustandsänderungen eines Modells während (Online-Animation) oder nach der Simulation (Offline-Animation)**
- **Hauptgründe für die Anwendung der Animation:**
 - **Anschauliches Debugging des Simulationsmodells, Fehlererkennung und -lokalisierung**
 - **Unterstützung der Modellvalidierung**
 - **Vermittlung der Funktionen und der Logik des Simulationsmodells**
 - **Anschauliche Präsentation von Simulationsergebnissen**
 - » **z.B. Visualisierung von Engpässen**

Zum Stellenwert der Animation

- **Animation wichtiges Hilfsmittel zur Kommunikation mit dem Anwender**
- **Unterscheidung zwischen Online- und Offline-Animation**
 - s. Kosturiak, Simulation von Produktionssystemen, Abb. 107, S.159
- **Unterschiedliche Ansätze zur Vorbereitung einer Animation im Rahmen von Simulationssystemen**
 - Integration in Modellbausteine
 - » Vorbereitung während des Modellaufbaus und danach unmittelbar startbereit
 - » Beispiele: SIMPLE++, SIMFACTORY II.5

Animation

- **Ansätze zur Vorbereitung einer Animation im Rahmen von Simulationssystemen, Forts.**
 - Trennung der Animation und Grafik von Modellaufbau und Modellbausteinen
 - » Modellaufbau in Modellierungsschicht
 - » Vorbereitung der Animation mit grafischem Editor
 - » Importfunktion für grafische Dateien mit Bildern aus CAD-Systemen als statischer Animationshintergrund
 - » Beispiel: SIMAN/ CINEMA
 - Animation mit externen Systemen über Schnittstelle
 - » Beispiel: CAD Motion

Funktionalität heutiger Animationssysteme

• Typische Funktionen

- Vollständige grafische Unterstützung bei der Vorbereitung der statischen und dynamischen Animationselemente
 - » Hintergrund, bewegliche Elemente
 - » unterschiedliche Farben oder Symbole zur Darstellung verschiedener Systemzustände
- Mehrere Animationsfenster für die Abbildung der unterschiedlichen Modellbereiche
- Stufenweise Veränderung der Animationsgeschwindigkeit und verschiedene Ansichtsformen
 - » Zoom
- Online-, Echtzeit-Animation
- Bibliothek der grafischen Symbole
 - » Standardsymbole, benutzerdefinierte Symbole
- CAD-Schnittstellen (z.B. DXF, IGES,..)

Funktionalität heutiger Animationssysteme

• Typische Funktionen

- Playback
 - » Aufzeichnung einer Situation und nachträgliches Abspielen und Analyse
 - » Zeitliches Vorwärts- und Rückwärtsspringen ohne neuen Simulationslauf
- Interaktive Ablaufsteuerung
 - » Ein- und Ausschalten
 - » Schritt für Schritt- Steuerung
 - » Snapshots - Festhalten des gegenwärtigen Systemzustandes

• Bewertung der Animation

- Wertvolle Hilfestellung bei Modelltest und -validierung
- Gutes Kommunikationsmittel mit Anwendern in Simulationsprojekt
- Aber: Simulationsergebnisse auf Basis einer fundierten statistischen Auswertung wiederholter Läufe, nicht begrenzter visueller Eindrücke

4.4 Anforderungen an moderne Simulationssysteme

- **Unterstützung bei der Datenvorbereitung und -eingabe**
 - Integrationsfähigkeit mit externen DV-Systemen:
 - » PPS-Datenbanken, Betriebsdatenerfassung, Arbeitsplanung
 - Schnittstellen zu CAD-Systemen
 - » Übernahme von Layouts und der grafischen Symbole für die Animation
 - Statistische Analysefunktionen für die Eingabedaten
 - » Chi-Quadrat-Test für Eingabevertelungen
 - Funktionen zur Überprüfung der Datenvollständigkeit
- **Unterstützung bei der Modellerstellung**
 - Grafischer Modellentwurf aus Grundbausteinen
 - Veränderliche Eigenschaften der Modellbausteine
 - Objektorientierte Entwicklung benutzerspezifischer Bausteine
 - Reichhaltige Unterstützung bei der Fehlersuche
 - Hierarchische Strukturierungsmechanismen

Anforderungen an moderne Simulationssysteme

- **Unterstützung bei der Modellimplementierung**
 - Leistungsfähige Modellierungssprachen mit hoher Flexibilität und Ausdruckstärke
 - ggf. automatische Umsetzung des grafisch spezifizierten Modells in ablauffähiges Simulationsprogramm
 - Schnittstellen zu verbreiteten höheren Programmiersprachen
 - Kopplungsmöglichkeiten vorhandener, ausführbarer Modellbausteine
 - Verwaltungsfunktionen für Modellbausteine
- **Unterstützung bei den Simulationsexperimenten**
 - Berechnung der Simulationszeit (z.B. aus dem Konfidenzintervall)
 - Unterstützung bei der Variantenbildung und Parameteränderung
 - Abschneiden der Anlaufphase
 - Statistische Verfahren zur "Simulationsoptimierung"

Anforderungen an moderne Simulationssysteme

- **Unterstützung der Simulationsdurchführung**
 - Interaktion mit dem Modell während der Simulation
 - » Interaktive Änderung der Modellparameter
 - Steuerung von Simulationsläufen
 - Online- und Offline-Animation
 - Interaktiver Debugger für Überwachung der Systemvariablen und Ablaufkontrolle
- **Auswertung der Simulationsergebnisse**
 - Breites Spektrum von Standardstatistiken
 - Statistische Auswertungsverfahren für benutzerdefinierte Analysen
 - Grafische Auswertungsfunktionen
 - Schnittstellen zu Tabellenkalkulation, Statistikpaketen, Standard-Datenbanken, Grafiksystemen
- **Zusammenfassung der Anforderungen im Überblick**
 - s. Kosturiak, Simulation von Produktionssystemen, Abb. 106, S.158
 - » “Expertensystem” im Sinne einer Benutzerunterstützung

4.5 Auswahlkriterien für Simulationssoftware in der Praxis

- **Großes kommerzielles Angebot an Simulationssoftware**
 - ca. 200 Produkte, davon 50% für Produktionssysteme einsetzbar
- **Modellierungskonzept**
 - Diskret/ Kontinuierlich; Weltbilder
 - Softwaregeneration
 - Simulations-/ Programmiersprache
 - Modellstrukturierung
 - » Hierarchisierungsmechanismen
 - Bausteinkonzept
 - Abbildungsgenauigkeit, Flexibilität
 - Objektorientierung
 - Anschaulichkeit
 - » Grafische Modellierung...

Auswahlkriterien für Simulationssoftware in der Praxis

- **Anwendungsgebiet**
 - Fertigung, Materialfluss, Logistik, Informationsfluss...
- **Leistungsaspekte der einzelnen Systemfunktionen**
 - Modellierungselemente und ihre Attribute
 - Statistik und stochastische Verteilungen
 - Experimentierunterstützung
 - Fehlersuche...
- **Benutzungsoberfläche**
 - Grafische Unterstützung, Fenstertechnik
 - Kontextsensitive Hilfefunktion...
- **Anforderungen an den Anwender**
 - Erforderliche Simulationserfahrungen
 - Spezielle Programmier-/ Softwarekenntnisse
 - Einarbeitungszeit

Auswahlkriterien für Simulationssoftware in der Praxis

- **Preis und Betriebskosten**
 - Kostenspanne für Simulationssoftware von 1000.- bis über 100.000 DM
 - Höhere Kosten für leistungsfähigere Versionen auf Unix-Workstations als PC-Versionen
 - Betriebskosten, vor allem Personalkosten
 - » i.d.R. wichtiger als Beschaffungskosten
- **Hardware-/ Softwareanforderungen**
 - Rechnerplattform, Betriebssystem, Bildschirmauflösung, Benutzungsschnittstellen (X-Windows), ...
- **Ergebnisse und ihre Darstellung**
 - Statistiken, Grafiken, Benutzerdefinierte Ausgabemöglichkeiten
- **Animation**
 - Online, Offline, 2D/ 3D

Auswahlkriterien für Simulationssoftware in der Praxis

- **Schnittstellen**

- CAD, ASCII, SQL-Datenbanken, Tabellenkalkulation (EXCEL), Programmierschnittstellen (C++)

- **Dienstleistungen, Verbreitungsgrad, Referenzen, Weiterentwicklung**

- Dokumentation, Wartung, Installationsanzahl
 - » typischerweise zwischen 10-100
 - » bei lang etablierten Produkten 1000-10.000 (viele Unis)
 - » **Beispiele: SLAM, SIMAN**
- Benutzerschulung/ -unterstützung, Beratung,
- Referenzen von anderen Anwendern
- Verfügbarkeit von Testinstallationen/ Demos
- Weiterentwicklung / neue Versionen
 - » Checkliste für die Auswahl einer Simulationssoftware:
s. Kosturiak, Simulation von Produktionssystemen, Abb. 110, S.164-166

4.6 Ausgewählte Beispiele für Simulationssoftware

4.6.1 Das Simulationspaket DESMO

- **Entwicklung des Simulationspaketes DESMO am FBI**

- Discrete Event Simulation Modelling
- DEMOS von *G.M. Birtwistle, Univ. of Calgary* als Vorbild
- DESMO in Modula-2 als erste lauffähige Version am FBI
 - » *Bölckow, et.al.* (s. Diskrete Simulation, Kap. 7)
- Weitere (Teil-) Implementierungen in
 - » Smalltalk, Oberon, C++
 - » Aktuell: Java, Delphi
- Objektorientiertes DESMO in C++
 - » *Schniewind (1998)*
 - s. T. Schniewind, Diplomarbeit "Entwurf und Implementierung eines Simulators in C++"
 - » **DESMO C++ = Klassenbibliothek für C++**

Funktionalität des Simulationspaketes DESMO

- **Alternative Modellierungsstile**
 - Ereignisorientiert
 - Prozessorientiert
 - » Transaktionsorientiert
 - Aktivitätsorientiert
 - Kombinierbarkeit verschiedener Modellierungsstile in einem Modell
 - » Beispiel: Ereignis “Ausfall einer Maschine” in einem prozessorientierten Modell des Fertigungssystems

=> Einsatz in der Simulationsausbildung
- **Prozessoperationen**
 - *Hold, Activate, ReActivate*
 - *Passivate, Cancel, Interrupt*
- **Erstellung wiederverwendbarer Modelle, die frei kombinierbar sind**

Funktionalität des Simulationspaketes DESMO

- **Elemente des objektorientierten Ansatzes**
 - Klassen / Objekte / Vererbung / Polymorphismus
 - Kommunikation zwischen Objekten / Entities über Nachrichtenaustausch
 - Definition von Systemobjekten als Modellentities
 - Statistikobjekte (z.T. implizit)
- **Höhere Modellierungskonstrukte**
 - Modellierung auf höherem Abstraktionsniveau (problemorientierter)
 - Ressourcenwettbewerb
 - Produzenten-/ Konsumentenbeziehungen
 - Direkte Prozesskooperation
 - Bedingtes Warten
 - Unterbrechung und Verdrängung von Entities

Funktionalität des Simulationspaketes DESMO

- **Testhilfen und Fehlerbehandlung**
- **Deadlocküberwachung**
 - Prüfung bei wechselseitiger Belegung von Ressourcen
- **Warteschlangenmanipulationen**
 - explizite / implizite Warteschlangen
 - Einreihung/ Entnahme/ Suche mit Bedingungen
- **Statistische Funktionen**
 - Zufallszahlenerzeugung
 - Statistische Verteilungen
 - » Gleich-, Exponential-, Normal-Verteilung
 - » Erlang-Verteilung, empirische Verteilung
- **Datensammelobjekte**
 - mit automatischem Update/ ohne automatischem Update

Funktionalität des Simulationspaketes DESMO

- **Beispiel für ein Datensammelobjekt:**
 - Definition einer Berechnungsfunktion

```
class WaitingTime: public ValueSupplier {
public:
    virtual double Value () const {...};
} waitingTime;
```
 - Variablendeklaration und -definition

```
Tally CustomerWaitingTime (ownerModel, waitingTime);
```
 - Explizite Aktualisierung des Datensammelobjektes

```
CustomerWaitingTime,Update();
...
CustomerWaitingTime,Update();
```
- **Ausgabereports**
 - Automatische Standardreports / benutzerdefinierte Ergebnisreports

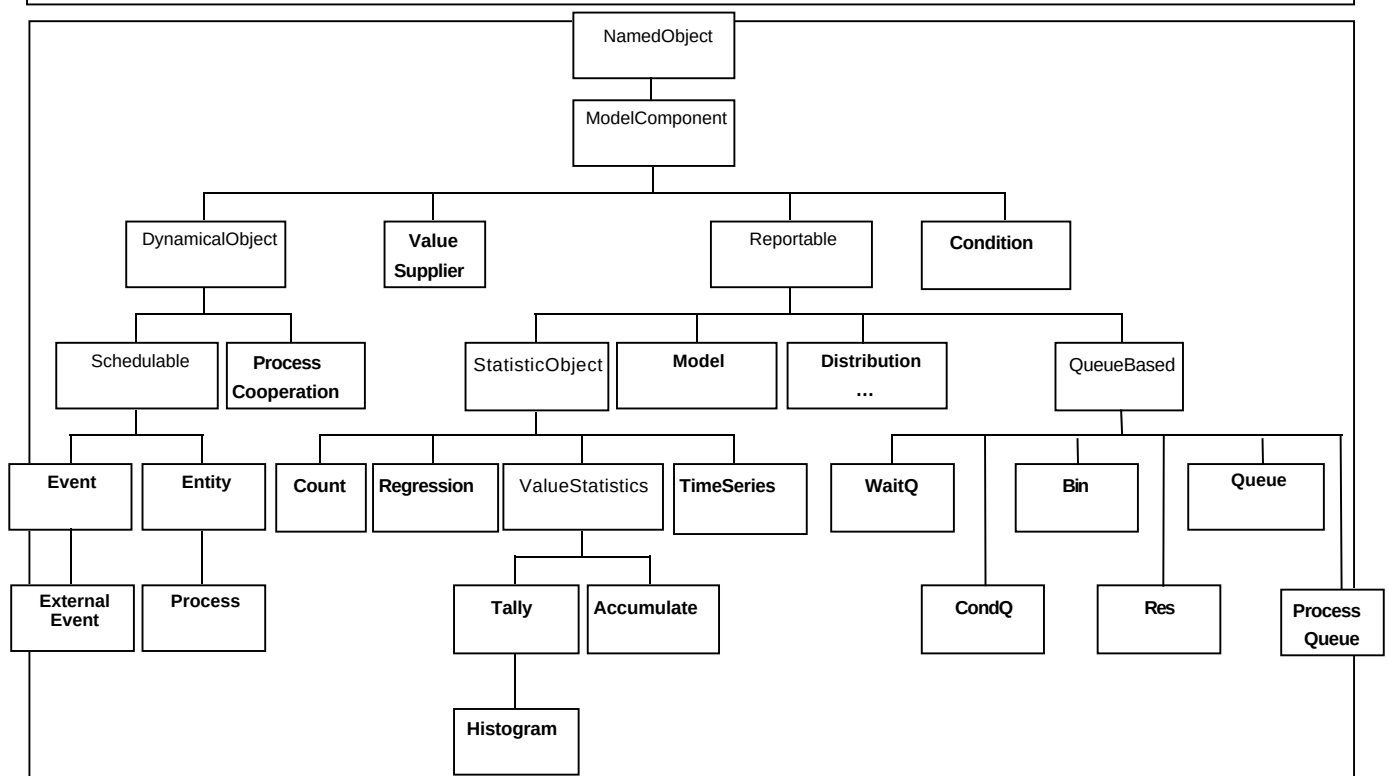
Funktionalität des Simulationspaketes DESMO

- **Modellentwicklung unabhängig von den Simulationsexperimenten**
 - Trennung der Modelle von den Experimenten
 - » Unterscheidung von Modell(-komponenten)-Objekten der Klasse *Model Component* und von Experiment-Objekten der Klasse *Experiment*
 - Modelle als Komponenten anderer Modelle
 - » Wiederverwendbarkeit, Unter-/ Submodelle, Modellhierarchien
 - Alle zum Modell gehörenden statitischen Komponenten im Modell gekapselt und mit ihm erzeugt
 - Zur Durchführung eines Experimentes mit dem Modell Anlage eines neuen Experiment-Objektes der Klasse *Experiment* , das Modell initialisiert und mit Parametern versorgt
 - Beliebige Wiederholung der Erzeugung, Durchführung und Vernichtung eines Experimentes ohne Abruch und Neustart des Simulationsprogramms

Funktionalität des Simulationspaketes DESMO

- **Simulationsexperimente**
 - Klasse *Experiment* mit Methoden zum Starten, Unterbrechen und Fortfahren von Experimenten
 - Methode zur Steuerung der Simulationsausgabe
 - Erzeugung eines Experimentes mit Vergabe eines Namens, der für Ausgabedatei verwendet wird
- **Aufbau der Klassenbibliothek zur zeitdiskreten Simulation**
 - Klassendiagramm
 - » Fett gedruckte Klassen zur direkten Nutzung oder Ableitung neuer Klassen
 - » Andere Klassen zur Zusammenfassung gemeinsamer Aspekte/ Funktionalität, die in abgeleiteten Klassen genutzt werden

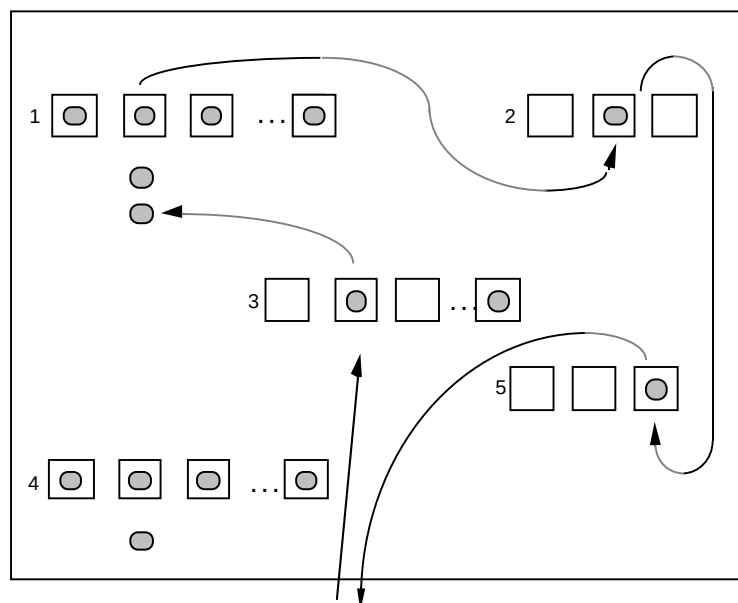
DESMO C++ - Klassenstruktur



Ein Simulationsbeispiel: Fertigungssystem in DESMO C++

• Modellierung des Fertigungssystems mit DESMO C++

- vgl. Kap. 2.5 + 2.6
- Engpassanalyse
- 5 Maschinengruppen mit identischen Maschinen
- 3 Jobtypen
- Bestimmung von
 - » Verweilzeit
 - » Maschinen- auslastung



Ein Simulationsbeispiel: Fertigungssystem in DESMO C++

- **Einrichtung einer allgemeinen Modellklasse *JobShop* mit Gemeinsamkeiten unterschiedlicher Versionen**
 - Darstellung der Routen durch den Maschinenpark
- **Transaktionsorientierte Version des Fertigungsmodells**
 - Aufträge als aktive Modell-Entities definiert, die Bedienungsnetz von Maschinenegruppen gemäss ihrer Jobtypen durchlaufen
 - Transaktionen = Jobs
 - Maschinengruppen = passive Komponenten
 - » Repräsentation als Ressourcen bestimmter Kapazität
 - Der Ablauf eines Auftrages in Klasse *Life Cycle* beschrieben
- **Ausgabe des Simulationsreports**

Ein Simulationsbeispiel: Fertigungssystem in DESMO C++

```
class Job : public Process
{ // Auftraege als aktive Modell-Entities
  public:
      Job (JobShopTrans& owner, unsigned jobType)
      :   Process      (owner, "Job"),
        type          (jobType),
        route         (&owner.GetRoute (type)),
        arrivalTime   (0.0)
      {};

      SimTime ArrivalTime() const { return arrivalTime;}

      virtual void   LifeCycle();

  private:
      unsigned      type;
      Route*        route;
      SimTime       arrivalTime;
};
```

Ein Simulationsbeispiel: Fertigungssystem in DESMO C++

```
void Job::LifeCycle() {
    JobShopTrans& jobShop = (JobShopTrans&)GetModel();
    // Nachfolger erzeugen:
    jobShop.CreateJob();

    arrivalTime = CurrentTime();

    Route* r = route;
    while (r)
    {
        MachineGroup& mGroup = r->GetMachineGroup();
        // Maschine exklusiv belegen:
        mGroup.Acquire (1);
        // Bearbeitung:
        Hold (mGroup.SampleServiceTime (type));
        // Maschine freigeben:
        mGroup.Release (1);
        r = r->Next();
    }
    jobShop.UpdateJobDelay (type);
    DeleteOnTermination();
}
```

Fertigungssystem in DESMO C++ Simulations - Report

```
*****
*                               *
*               Experiment: JobTrans                               *
*               Report                                             *
*                               *
*****

*****
*                               *
*               Model: JobShopTrans                               *
*                               *
*****

Description:
Job-Shop-Model (transaction oriented)
JobShopTrans reset at: 0.000

Distributions
Title          (Re)set  Obs Type Parameters      Seed
-----
Job Type        0.000  11631 I-Empirical      0          0.3000    33427485
                  1          0.8000
                  2          1.0000

InterArrival    0.000  11630 Neg-Expon.      0.2500      22276755
Service 1 1     0.000   3513 k-Erlang      0.6000      46847980
Service 2 1     0.000   5793 k-Erlang      0.8000      43859043
Service 3 1     0.000   2285 k-Erlang      0.7000      64042082
Service 1 2     0.000   3486 k-Erlang      0.8500      44366385
:

Queues
-----
Title          (Re)set  Obs   Qmax   Qnow   Qavg.  Zeros  avg.Wait
-----
Mach Group 1   0.000  11591   39     5     8.130  1126   2.047
Mach Group 2   0.000   5773   67    52    22.190  202   10.707
Mach Group 3   0.000  11590   11     0     0.711  6732   0.179
Mach Group 4   0.000   8081   58     4    14.490  612   5.235
Mach Group 5   0.000   5770   16     1     1.908  1466   0.966
```

Fertigungssystem in DESMO C++ Simulations - Report

Accumulates

Title	(Re)set	Obs	Mean	Std.Dev	Min	Max
Queue Len 1	0.000	54427	8.130	7.283	0.000	39.000
Queue Len 2	0.000	54427	22.190	16.326	0.000	67.000
Queue Len 3	0.000	54427	0.711	1.476	0.000	11.000
Queue Len 4	0.000	54427	14.490	12.880	0.000	58.000
Queue Len 5	0.000	54427	1.908	2.615	0.000	16.000

Tallies

Title	(Re)set	Obs	Mean	Std.Dev	Min	Max
Ttl Job Dly	0.000	11558	14.578	43.340	0.711	1220.996
Job Delay 1	0.000	3484	16.294	51.631	1.067	1220.643
Job Delay 2	0.000	5792	9.906	23.807	0.711	395.581
Job Delay 3	0.000	2282	23.813	62.148	1.992	1220.996

Bewertung des Simulationspaketes DESMO

- **DESMO Simulationssoftware auf der Ebene der Simulationspakete**
- **Simulationspaket hoher Funktionalität und Flexibilität**
- **Objektorientierter Ansatz**
- **Simulationsprogramme in streng typisierter, objekt-orientierter Programmiersprache**
 - Gut strukturierte Programme
- **Anwendungssicherheit**
 - Reduzierung der Fehlermöglichkeiten durch Datenkapselung
- **Unterstützung verschiedener Abstraktionsebenen und mehrerer Modellierungsstile**

Bewertung des Simulationspaketes DESMO

- **Wiederverwendbarkeit von (Teil-) Modellen**
 - Modellhierarchien aus Modellkomponenten
- **Unterstützung der Trennung von Modell und Experiment**
 - Eigenständige Objekte
 - Mehrere Simulationsobjekte mit einem Modell ohne Programmneustart möglich
- **DESMO gutes didaktisches Hilfsmittel zur Erlernung der wichtigen Simulationskonzepte**

4.6 Ausgewählte Beispiele für Simulationssoftware

4.6.2 Das Simulationssystem EXTEND

- **EXTEND = Grafisches Modellierungswerkzeug für Macintosh- oder Windows- PC-Plattformen**
- **Offenes Simulationssystem für Systemerweiterungen auf Benutzerebene**
 - Grafische Modellerstellungskomponente und C-ähnliche Modellierungssprache ModL
- **Grundsätzlicher Ansatz: Bibliotheksbasiertes Blockkonzept**
 - Beim Modellaufbau Entnahme vorgefertigter Modellbausteine (= Blöcke) aus mitgelieferter Standardblockbibliothek
 - Eigene Erstellung von Bibliotheken für spezielle Anwendungen möglich
 - » Modellierungs-Programmiersprache ModL

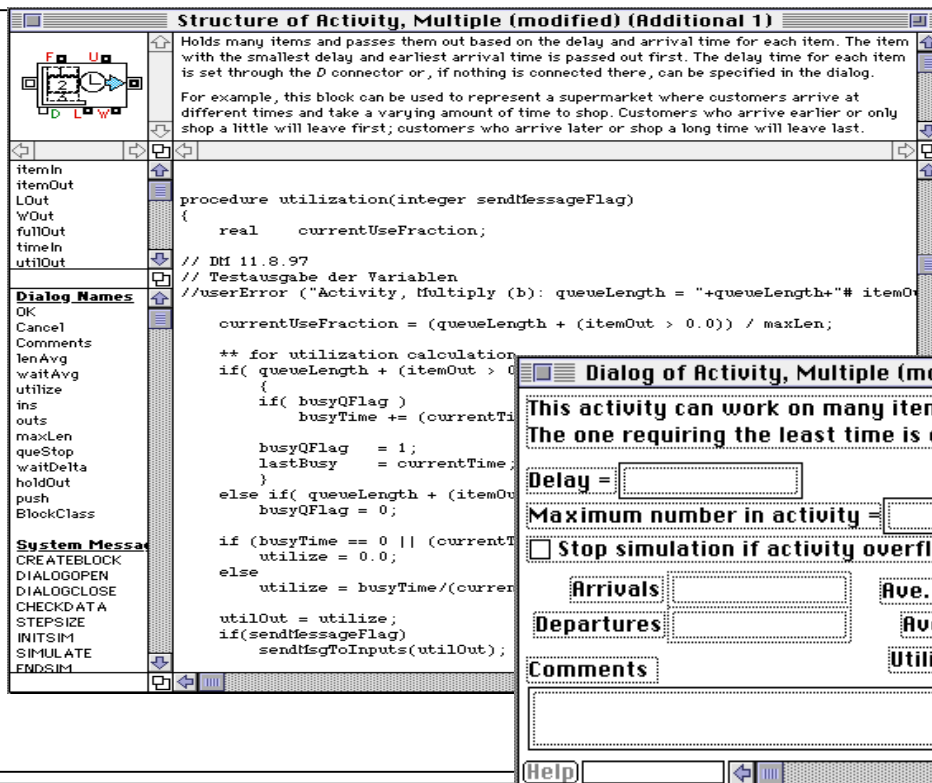
Das Simulationssystem EXTEND

- **EXTEND mit Bibliotheken für diskrete, kontinuierliche und kombinierte Modelle**
- **Unterstützung grafischer, hierarchischer Modellierung**
 - Modellaufbau durch menü- und fenstergesteuerte Auswahl von Blöcken aus der Bibliothek und Platzierung in Modelldiagramm
 - Blöcke als grafische Repräsentationen von Aktionen oder Prozessen
 - Jeder Block mit Konnektoren für Verbindungslinien mit anderen Blöcken
 - » Anschlussstellen
 - Informationsfluß zwischen Blöcken über Konnektoren und Verbindungslinien
 - Verarbeitung der Informationen durch zugehörige Programmskripts
 - » Eigene Modellierungssprache ModL verfügbar
 - Hierarchische Blöcke mit beliebig geschachtelten Submodellen

Das Simulationssystem EXTEND

- **Abspeicherung der Blöcke in separaten Bibliotheken unabhängig von zugehörigem Modell**
 - Import von Blöcken in Modellprogramm möglich
 - Jeder Block mit Programmskript, Icon, Konnektoren, Dialogfenster und Dokumentation
 - Dialogfenster zur Eingabe von Parameterwerten und zur Verfolgung von Simulationsergebnissen
 - Entwurf von Dialogfenstern mit Knöpfen, Textfeldern, Auswahlfeldern,... durch Benutzer
- **Benennung von Blockverbindungen**
 - "Named connections" ohne Verbindungslinien
 - Vermeidung von langen, unübersichtlichen Verbindungslinien

Das Simulationssystem EXTEND



Aufbau eines EXTEND-Blocks

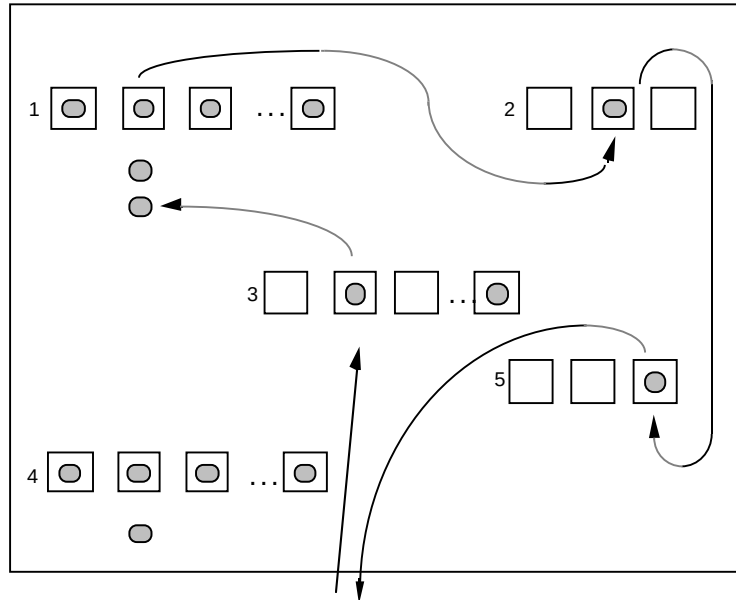
Das Simulationssystem EXTEND

- **Ausgabeblocks zur Auswertung von Simulationsergebnissen**
 - Ausgabeplots, Histogramme, Zeitreihen...
- **Animationsfunktion**
 - Menügesteuerte Anforderung durch Benutzer
 - » Beispiel: Blinkende Blöcke
 - Entwicklung eigener Animationen zu den Blöcken über Programmfunktionen
 - » Zuweisung von Farben, Formen, Bildern und Filmen an Blöcke
- **Einfache Schnittstellen zum Import / Export beliebiger Daten über Blöcke**
 - Ausgabe von Simulationsergebnissen in Datei
 - Einlesen von Daten während der Simulation aus Datei / Datenbank

Simulationsbeispiel mit EXTEND - Fertigungssystem

• Modellierung des Fertigungssystems mit EXTEND

- vgl. Kap. 2.5 + 2.6
- Engpassanalyse
- 5 Maschinengruppen mit identischen Maschinen
- 3 Jobtypen
- Bestimmung von
 - » Verweilzeit
 - » Maschinen-
auslastung



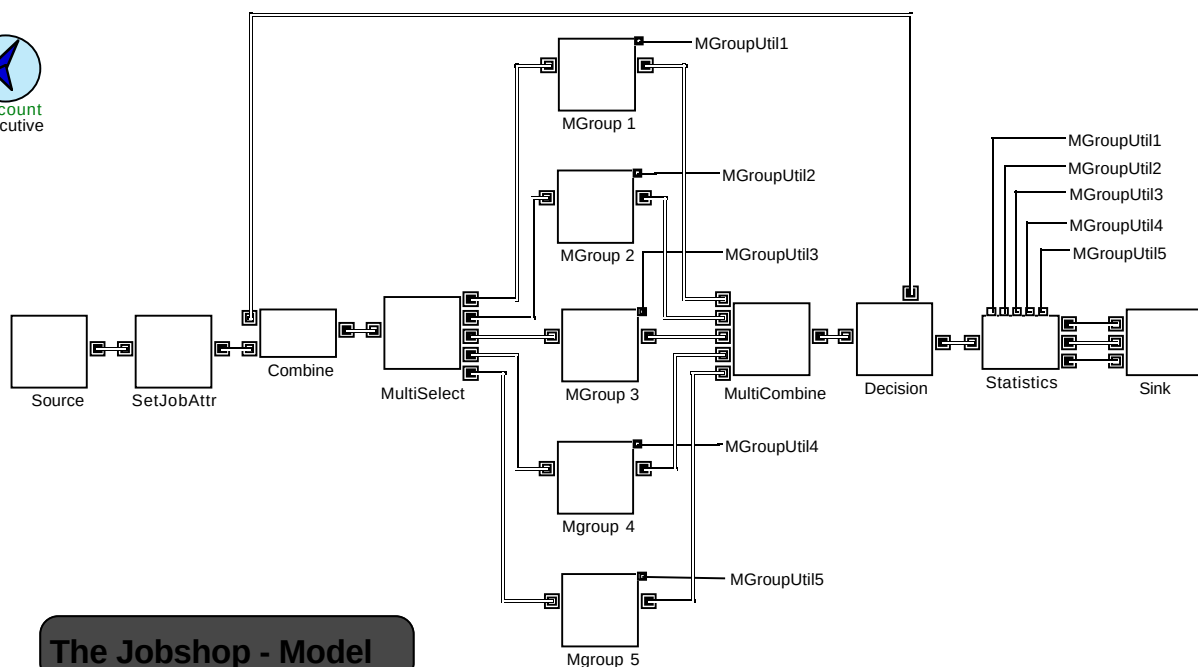
Simulationsbeispiel mit EXTEND - Fertigungssystem

- Aufbau des Fertigungsmodells mit grafischen Standardblöcken der EXTEND-Bibliothek, die parametrisiert werden
- Jobshop-Modell auf oberster Ebene mit benutzerdefinierten Blöcken:
 - Generierung der Jobs durch *Source-Block*
 - *SetJobAttr* - Block zum Setzen der Attribute der Jobs
 - *MultiSelect*- Block zur Auswahl der nächsten Maschinengruppe in Abhängigkeit der Bearbeitungsreihenfolge
 - Repräsentation der Bearbeitung der Aufträge durch Maschinen-
gruppen über Blöcke *MGroup 1,... MGroup 5*
 - Block *MultiCombine* zur Zusammenführung der Objekt-Ströme aus den Maschinengruppen
 - Block *Decision* zur Auswahl von fertigen oder weiter zu bearbeitenden Jobs

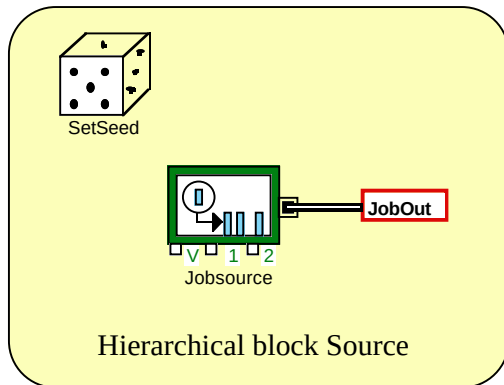
Simulationsbeispiel mit EXTEND - Fertigungssystem

- **Jobshop-Modell auf oberster Ebene mit benutzerdefinierten Blöcken, Forts.:**
 - *Statistics*- Block zur Fortschreibung der Statistiken für die einzelnen Maschinengruppen und Jobtypen
 - *Sink*- Block für die Entfernung der fertigen Jobs aus dem System
- **Realisierung der benutzerdefinierten Blöcke auf oberster Hierarchieebene durch parametrisierte Standardblöcke auf der darunter liegenden Ebene**
 - keine Programmierung in der Modellierungssprache ModL erforderlich
 - Beispiel: Hierarchischer Block *Source* repräsentiert durch Standardblock *Generator*, der im Modell mit *Jobsource* bezeichnet wird
- **Ergebnisdarstellung durch *Plotter*- Block in *Statistics***

Simulationsbeispiel mit EXTEND - Fertigungssystem

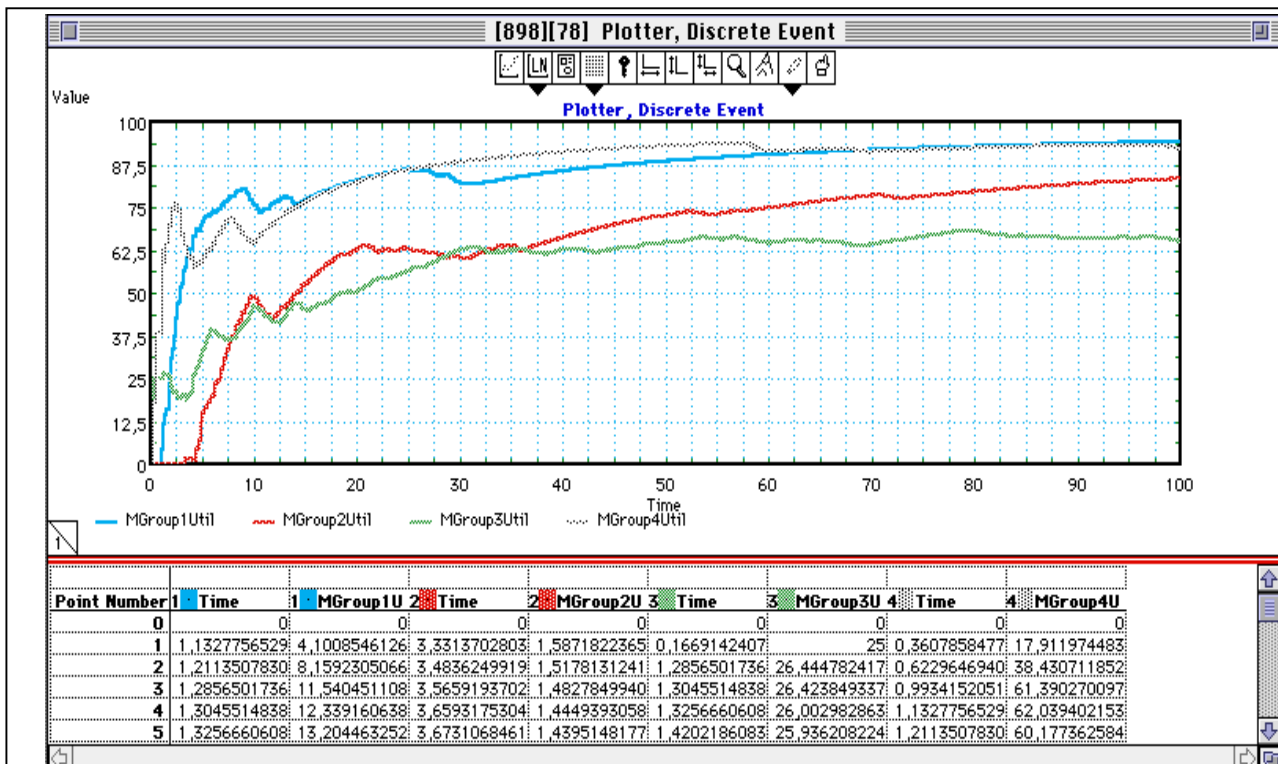


Simulationsbeispiel mit EXTEND - Fertigungssystem



Generator dialog ("Jobsource")

Simulationsbeispiel mit EXTEND - Fertigungssystem



Bewertung von EXTEND

- **Simulationswerkzeug auf der Ebene der Simulationssysteme**
- **EXTEND sehr offenes und flexibles Simulationssystem**
- **Grafische Modellierung mit vorgefertigten Modellbausteinen (Blöcken)**
 - Standardblöcke und benutzerdefinierte Blöcke in modellunabhängigen Bibliotheken
- **Hohe Erweiterbarkeit für spezielle Anwendungsbereiche**
 - Benutzerdefinierte Bibliotheken
- **Hierarchische Modellierung**
- **Animationsfunktionen**

Bewertung von EXTEND

- **Fensterorientierte Modelldialoge**
 - Benutzerspezifisch zu definieren
- **Eigene Modellprogrammiersprache ModL**
 - C-verwandte Sprache zur Programmierung der Modellbausteine
- **Datenbankschnittstelle möglich**
- **EXTEND gut geeignet als Basissystem zur Realisierung eigener Simulatoren für spezielle Anwendungsgebiete**

4.6.3 Übersicht über weitere Softwareprodukte für die diskrete Simulation

- **Tabellarische Produktübersicht vorrangig für Produktionssysteme**
 - s. Simulation von Produktionssystemen (1995), S. 167-180
 - » Systeme: ARENA, EXTEND, GPSS/H, GPSS/PC, MicroSaint, MOSYS, Siman V/ Cinema V, SIMFACTORY II.5, SIMPLE ++, SIMPLEX II, SLAMSYSTEM, TAYLOR II, WITNESS
- **Softwarevergleich an Hand von Beispielmodellen**
 - Vorgegebene, leicht verständliche Beispielmodelle mit spezifischen Anforderungen an Modellierung und Simulationsexperimente werden von Softwareanbietern mit ihrer Simulationssoftware gelöst und in fest definiertem Format knapp dokumentiert.
 - Insgesamt 12 Beispielmodelle mit ca. 150 eingesendeten Lösungen
 - Durchführung fortlaufend ab 1991 durch die deutsche ARGESIM
 - » Publikation in Simulation News Europe der EUROSIM
 - » WWW-Server: <http://www.eurosim.org/>

Übersicht über weitere Softwareprodukte für die diskrete Simulation

- **Aktuelle Marktübersicht zu Simulationssoftware für zeitdiskrete Systeme an Hand von Vergleichskriterien**
 - 1997 Simulation Software Survey des amerikanischen Institutes for Operations Research and the Management Sciences
 - Vergleichskriterien:
 - » Anbieter, Grafische Modellierung, Schätzfunktionen für Eingabeverteilungen, Wiederverwendbarkeit von Submodellen, Unterstützung für die statistische Analyse der Ausgabedaten, interaktive Debugunterstützung, typische Anwendungsbereiche, Speicherbedarf, Betriebssystemplattformen, Kosten, wichtige Neufunktionen
 - WWW-URL: // <http://lionhrtpub.com/orms/>