

# **Evaluierung und Analyse integrationsfähiger Simulations Engines für die Entwicklung komplexer und detaillierter Simulationsmodelle**

Markus Speckle, Matthias Gruber, Martin Saler

markus.speckle@v-research.at, matthias.gruber@profactor.at, martin.saler@v-research.at  
V-Research GmbH / Industrielle Forschung und Entwicklung / Technische Logistik  
Stadtstraße 33 / A-6850 Dornbirn

## **Kurzfassung**

Für die Durchführung von Simulationsstudien in Produktion und Logistik steht den Simulationsexperten eine Vielzahl von Simulationswerkzeugen zur Verfügung. Die Anforderungen an diese Simulationswerkzeuge und das Expertenwissen der Simulationsexperten verändert sich ständig. Für Simulationswerkzeuge besteht immer öfters die Anforderung der Integrationsfähigkeit hinsichtlich bestehender Umgebungen und setzt daher besondere Anforderungen an die Schnittstellenfunktionalität dieser Werkzeuge. Sofern das Werkzeug in einer bestehenden Umgebung im Hintergrund für die Ausführung bestehender Modelle verwendet und die Auswertung der Ergebnisse durch andere Softwarekomponenten übernommen wird, kann die Verwendung des Simulationswerkzeuges auch als Simulations Engine bezeichnet werden. V-Research realisierte auf dieser Basis bereits erfolgreich mehrere Simulationsstudien und ermöglicht dem Planungsexperten durch die eigens entwickelte *Anwendungsplattform Simulation (AWPS)*, projektspezifische Modellerstellungs- und Auswertungsoberflächen bereit zu stellen. Für solche Umgebungen würden sich weiters aber auch Simulationsbibliotheken als Simulations Engine anbieten, welche sich verstärkt im Aufbau befinden [4]. Diesbezüglich wurde eine Marktanalyse mit verschiedenen Kriterien durchgeführt, welche zum Ziel hatte, die Eignung von Simulationswerkzeugen und Simulationsbibliotheken als integrierbare Simulations Engine für die Entwicklung von komplexen und detaillierten Simulationsmodellen zu erfahren.

## **1 Einleitung**

Während ab Beginn der 60er Jahre Simulationsstudien mit Simulationssprachen wie bspw. GPSS, SIMSCRIPT, ALGOL oder SIMULA durchgeführt wurden, entwickelten sich später daraus immer mehr visuelle Entwicklungsumgebungen (GPSS/NORDEN, SIMAN). Ab Mitte der 80er Jahre entwickelte sich der Trend bis heute zu Simulationsumgebungen mit grafischen Benutzer- und Modellierungsoberflächen, 2D/3D Animationen und anderen Visualisierungskomponenten [1]. Vor allem durch die verfügbaren Möglichkeiten zur Animation und Visualisierung wurde die Simulation immer populärer [2].

Weiters ist auch eine Entwicklung hinsichtlich der Erstellung und Verwendung von Simulationsbibliotheken in den verschiedensten Anwendungsgebieten zu erkennen ([3], [4]). Eine der möglichen Faktoren für diese Entwicklung könnte die ständig präsente Anforderung zur Integration der Simulationswerkzeuge in eine bestehende Umgebung sein und die hohen Anforderungen seitens der automatischen Modellgenerierung [4].

Diesbezüglich werden in diesem Beitrag auf die Ergebnisse von [3] eingegangen, in welcher 52 Simulationswerkzeuge und 16 Simulationsbibliotheken hinsichtlich definierter Ausschluss- und Bewertungskriterien für die Integration in eine bestehende Umgebung (AWPS) analysiert wurden.

## 2 Simulationswerkzeuge und Simulationsbibliotheken

Simulationsbibliotheken unterscheiden sich von Simulationswerkzeugen einerseits durch die fehlende grafische Modellierungsoberfläche und fehlende integrierte Funktionalität zur grafischen Auswertung der Resultate. Weiters bieten Werkzeuge oftmals optisch ansprechende 2D/3D Visualisierungsmöglichkeiten, welche bei Bibliotheken ebenfalls zur Gänze fehlen. Für Bibliotheken werden weiters die Simulationsmodelle ausschließlich programmatisch durch Quelltext definiert. Beide Systeme umfassen jedoch wesentliche Simulationskonzepte zur Erstellung und Ausführung von Simulationsmodellen und bieten zu diesem Zweck eine Sammlung von vordefinierten Modellbausteinen, welche für die Modellierung der statischen und dynamischen Systemelemente dienen.

Der Begriff „Simulations Engine“ beschreibt grundsätzlich ein Simulationsinstrument, welches wesentliche Simulationskonzepte wie die Ereigniserzeugung und Ereignisverwaltung zur Verfügung stellt, aber auch Modellbausteine und die Modellverwaltung beinhaltet. Dabei steht die Integration in eine bestehende Umgebung im Vordergrund und setzt somit eine Softwareschnittstelle voraus. Eine grafische Modellierungsoberfläche oder Benutzeroberfläche ist nicht zwingend Bestandteil einer Simulations Engine. Dadurch soll es bestehenden Anwendungsumgebungen ermöglicht werden, Simulationsmodelle zu parametrisieren, auszuführen und die Ergebnisse selbst auszuwerten. Die Simulations Engine ist dabei als Subsystem eines Gesamtsystems zu verstehen. Der Planungsexperte (nicht Simulationsexperte) kann dadurch ohne Interaktion mit dem Simulationsinstrument (bspw. Bibliothek oder Werkzeug) verschiedene Simulationsmodelle aus der bestehenden Anwendungsumgebung heraus durchführen. Sofern eine Simulationsbibliothek oder ein Simulationswerkzeug geeignete Schnittstellen für die Integration zu Verfügung stellt, kann dieses als Simulations Engine verwendet werden.

Dieses Konzept wurde in der *Anwendungsplattform Simulation (AWPS)* von V-Research bereits erfolgreich für verschiedene detaillierte und komplexe Simulationsstudien umgesetzt (Vgl. [5] und [6]). Die Simulationsmodelle einer AWPS-Umgebung werden vor jedem Simulationslauf automatisch erzeugt. Dies bedeutet, dass das eigentliche Simulationsmodell aus den Eingabedaten (Parameter, Struktur- und

Prozessdefinitionen) des Planungsexperten und einer modellgenerierenden Logik besteht bzw. generiert wird. Die modellgenerierende Logik kann dabei auch als Basismodell verstanden werden. Für die AWPS-Umgebung entstanden dadurch besondere Modellierungsanforderungen, welche die Möglichkeiten der grafischen Modellierung überschreitet und ausschließlich programmatisch umgesetzt werden kann.

Bestehende und zukünftige Simulationsmodelle hinsichtlich der AWPS-Umgebung können als detaillierte und komplexe Modelle bezeichnet werden. Diese wurden weitgehend realitätsnah und detailliert umgesetzt, sodass die geforderten Gesamt- und Einzelprozesse eines Systems durch den Planungsexperten genau untersucht werden konnten (verzicht auf eine hohe Abstraktionsebene).

### **3 Definierte Ausschluss- und Bewertungskriterien für Simulations Engines**

Eine erste, grobe Marktanalyse hat sehr früh gezeigt, dass eine größere Menge an Werkzeugen bzw. Bibliotheken für die Produktions- und Logistik-Simulation verfügbar ist. Bei den Bibliotheken ist anzumerken, dass diese zum größten Teil nicht speziell für die Produktions- und Logistik-Simulation ausgelegt sind (wenige Modulbausteine). Dennoch war eine sehr hohe Modellierungsflexibilität beim Einsatz einer Bibliothek zu erwarten, welches die Implementierung der benötigten Modellbausteine erleichtert. Aus diesem Grund wurden Simulationsbibliotheken und Simulationswerkzeuge getrennt aufgelistet.

Als Hauptquellen für die Werkzeuge wurde die Marktanalyse von *Operations Research and the Management Sciences 2007* [7, 8] herangezogen und mit den Werkzeugen und Bibliotheken von [9] und [10] ergänzt. Nach Abschluss der Recherche standen 52 Simulationswerkzeuge sowie 16 Simulationsbibliotheken zur Verfügung. Diese Werkzeuge wurden anschließend hinsichtlich speziell definierter Ausschlusskriterien (Orientierung anhand der AWPS-Umgebung) überprüft und gegebenenfalls aus der Liste gestrichen, falls diese nicht erfüllt werden konnten.

Die Schwerpunkte bei den Ausschlusskriterien lagen vor allem in der Integrationsfähigkeit und Durchführung detaillierter und komplexer Modelle. Dies setzte Schnittstellenanforderungen für das Laden, Parametrisieren, Starten und Stoppen von bestehenden Simulationsmodellen voraus. Zudem wurde gefordert, dass eine integrierte Programmiersprache zur Modellentwicklung in den Werkzeugen zur Verfügung steht, welche beispielsweise das Erzeugen und Definieren von Modellbausteinen, Ressourcen, Verbindungen und Prozessabläufen ermöglicht (strukturelle Änderungen am Modell programmatisch ermöglicht). Als weiteres Ausschlusskriterium wurden aus Gründen der Effizienz auf Plug-In basierende Werkzeuge (bspw. Spreadsheet Simulatoren) verzichtet und gefordert, dass 50.000 Flussobjekte (Entitäten, Objekte, z.B. Paketsendungen) simultan simulieren werden können (davon 10% gleichzeitig aktive Objekte).

Die Kriterien für die Simulationsbibliotheken fokussierten sich hauptsächlich auf eine objektorientierte Implementierungssprache und eine vollständig dokumentierte Schnittstellenfunktionalität. Weiters wurde von den Bibliotheken die Unterstützung der

ereignisdiskreten Simulation, kommerzielle Nutzbarkeit und Erweiterbarkeit im Sinne der Vererbung vorhandener Klassen und Modellbausteinen vorausgesetzt.

Nach der Überprüfung auf die Ausschlusskriterien standen nunmehr vier Simulationswerkzeuge (Micro Saint Sharp, ExtendSim AT, Arena, Witness) und drei Simulationsbibliotheken (SiRO, Delsi, CSIM19) für die Bewertung anhand der definierten Kriterien (siehe unten), zur Verfügung. Ebenso wie die Ausschluss- wurden die Bewertungskriterien der Werkzeuge und Bibliotheken getrennt definiert und anschließend für die Bewertung der näher analysierten Systeme verwendet.

Im Folgenden werden nun die jeweiligen gewichteten Bewertungskriterien für die Simulationswerkzeuge und Simulationsbibliotheken dargestellt:

Bereichsblock und Kriterien	Gewichtung	
<b>Systemanforderungen</b>	<b>5%</b>	
Grafikkarten Anforderungen	100%	
<b>Schnittstellen</b>	<b>25%</b>	
Datenbankschnittstellen	20%	
Dateizugriff (IO Operationen)	5%	
Interne und externe API Funktionalität	45%	
Qualität der API Dokumentation	30%	
<b>Bedienung und allg. Funktionalität</b>	<b>10%</b>	
Benutzerfreundlichkeit der Oberfläche	20%	
Modularisierung	20%	
Parametrisierung	20%	
Vererbung	20%	
Geschwindigkeitsregulierung	20%	
<b>Programmierung und Entwicklungsumgebung</b>	<b>25%</b>	
Debug Funktionalität (Breakpoints, Watch Variablen)	30%	
Programmiersprache und dessen Funktionalität	30%	
Programmierunterstützung	20%	
Kollaborative Modellentwicklung (offenes Modellformat für Versionierung via CVS, SVN, etc.)	20%	
<b>Zusatzleistungen</b>	<b>10%</b>	
Schulungen (Beispiel Simulationen) / Trainings-Angebote)	50%	
Supportqualität	50%	
<b>Lizenzkosten</b>	<b>25%</b>	
Preis pro Entwickler-Lizenz (Professional Version)	30%	
Preis pro Simulations Engine (für AWPS Computation Unit)	70%	

**Bild: 1** Gewichtete Bewertungskriterien für Simulationswerkzeuge, Quelle [3]

Bereichsblock und Kriterien	Gewichtung	
<b>Systemanforderungen</b>	<b>10%</b>	
Zusatzanforderungen (zusätzlich benötigte Umgebung)		100%
<b>Schnittstellen</b>	<b>30%</b>	
Vererbung von bestehenden Typen		60%
Qualität der API Dokumentation		40%
<b>Allgemeine Funktionalität</b>	<b>15%</b>	
Funktionalität für Simulationen in Produktion und Logistik		60%
Mögl. zur Funktionalitätserweiterung durch Hersteller		20%
Testing der Bibliothek		20%
<b>Programmierung und Entwicklungsumgebung</b>	<b>10%</b>	
Programmiersprache (für die Modellierung)		50%
Freiheitsgrad in der Modellierung		50%
<b>Zusatzleistungen</b>	<b>10%</b>	
Schulungen (Beispiel Simulationen) / Trainings-Angebote)		60%
Supportqualität		30%
Kontinuierliche Weiterentwicklung		10%
<b>Lizenzkosten</b>	<b>25%</b>	
Preis pro Entwickler-Lizenz		30%
Preis pro Simulations Engine (für AWPS Computation Unit)		70%

**Bild: 2** Gewichtete Bewertungskriterien für Simulationsbibliotheken, Quelle [3]

## 4 Schlussfolgerungen der Analyse und Bewertung von Simulationsbibliotheken und Werkzeugen

Die durchgeführte Analyse der am Markt befindlichen Werkzeuge bezüglich verschiedener Ausschluss- und Bewertungskriterien führte zu dem Ergebnis, dass nur eine sehr eingeschränkte Anzahl an Simulationswerkzeugen zur Verfügung steht, die eine umfangreiche Schnittstellenfunktionalität für die Integration in eine bestehende Umgebung liefern und gleichzeitig auch für umfangreiche Simulationsmodelle geeignet sind. Auf Seiten der Simulationsbibliotheken stellte sich die Frage der Schnittstellenfunktionalität nur eingeschränkt, da diese für die Integration in bestehende Umgebungen im großen Umfang offen stehen und dafür bestens ausgelegt sind. Dennoch stellte sich heraus, dass spezielle Simulationsbibliotheken für die Anwendungsbereiche Produktion und Logistik im Gegensatz zu den Werkzeugen nur sehr eingeschränkt zur Verfügung stehen und somit viele Modellbausteine erst selbst entwickelt werden müssen. Aus diesen und weiteren Gründen wurden aktuell verfügbare Simulationsbibliotheken in die Analyse miteinbezogen und mit den Simulationswerkzeugen hinsichtlich der folgenden Punkte gegenübergestellt und analysiert:

**Unterschiede in der Modellierung:** Simulationswerkzeuge umfassen meist eine intuitive grafische Benutzeroberfläche für die Entwicklung der Simulationsmodelle. Dazu gehört häufig eine Vielzahl vordefinierter Modulbausteine, die für eine spezielle Modelllogik zur Verfügung stehen (Stapler, Roboter, Lagerflächen, etc). Diese müssen aber stets mit einer anwendungsspezifischen Logik in Form von Quellcode erweitert werden (Vgl. auch [1]), damit die gewünschte Modelllogik abgebildet ist. Die Anpassungen durch Quellcode fallen meist sehr komplex aus und erhöhen somit die

allgemeine Modellkomplexität. Dies führt somit zu einem schlecht strukturierten und unübersichtlichen Modell, da Quellcode und grafische Modellelemente an den unterschiedlichsten Stellen miteinander vermischt werden. Die Erstellung einer einheitlichen, übersichtlichen und systematischen Dokumentation des gesamten Systems ist fast nicht mehr möglich.

Dem gegenüber stehen die Simulationsbibliotheken, deren meist moderne objektorientierte Programmiersprachen eine hohe Modellierungsflexibilität durch die hierarchische Modellierung oder Techniken wie Namensräume, Verteilung auf Projektmodule und Datenkapselung erlauben. Der Aufwand für die Implementierung eines Modells ist gegenüber einem Simulationswerkzeug jedoch höher einzuschätzen, sofern wenig detaillierte und komplexe Logikanforderungen bestehen. Weiters standen in den untersuchten Bibliotheken nur wenige bis gar keine vordefinierten Modellbausteine für Produktion- und Logistik Simulationen zu Verfügung.

**Programmiersprachen für die Logikimplementierung:** Die Simulationswerkzeuge bieten meist einfache Programmiersprachen, die schnell erlernbar sind und keine Kompilierung vor dem Simulationslauf benötigen (sogenannte Interpretersprachen). Die Sprachen verwenden aber meist eine spezielle Syntax und macht auch bei vorhandenen Programmierkenntnissen bzgl. moderner Sprachen (Java, C++, .NET C#) eine Einarbeitung nötig. Weiters bieten diese Sprachen im Regelfall keine komplexen und effizienten Datenstrukturen wie dynamische Listen, Vektoren, Hash-Tabellen, Dictionaries, Properties und Attribute, welche somit eine laufzeitoptimierte Programmierung einschränken bzw. erschweren. Dennoch können teilweise externe Programmmodule oder Programmcode mittels COM, ActiveX, etc. Technologien verwendet werden, sodass auch moderne Programmiersprachen genutzt werden können. Diese Einbindung ist jedoch meist aufwendig und benötigt komplexe Kompilierschritte vor jedem Simulationslauf. Weiters ist in Folge dessen keine Modulüberschreitende Fehlersuche (Debugging) aufgrund unterschiedlicher Technologien mehr möglich.

Bibliotheken haben hinsichtlich der Programmiersprachen keine Einschränkung, da modernste objektorientierte Programmiersprachen zur Verfügung stehen und komplexe Kompilierschritte entfallen. Durch solche Sprachen steht ein Maximum an Modellierungsflexibilität zur Verfügung.

**Schnittstellen zur Integration in bestehende Umgebungen:** Die Analyse hat ergeben, dass nur wenige Werkzeuge eine Schnittstelle zur Integration bereitstellen und diese meist nur Funktionalitäten zur Parametrisierung, Starten und Stoppen von Modellen beinhalten. Wenige Hersteller bieten weiters auch Funktionalitäten für die Unterstützung der automatisierten und dynamischen Modellgenerierung an. Dies umfasst beispielsweise das Hinzufügen von beliebigen stationären, mobilen oder flussorientierten Objekten, die Erstellung von neuen Verbindungen.

Die Simulationsbibliotheken wiesen in den näher untersuchten Systemen bezüglich der Schnittstellenfunktionalität keine Einschränkungen auf und empfehlen sich daher aufgrund der Offenheit der Schnittstellen für eine automatisierte und dynamische Modellgenerierung.

**Entwicklungsumgebungen für die Modellogik:** In den Werkzeugen wurden großteils eingeschränkte Programmierunterstützungen ersichtlich. Die als de facto Standard geltenden Hilfsmittel gängiger Softwareentwicklungsumgebungen, stehen in den Werkzeugen bis auf sehr wenige Ausnahmen (MicroSaint Sharp, AnyLogic) nur stark eingeschränkt zur Verfügung. Hierzu zählen beispielsweise die automatische Vervollständigung von Quelltext, statische Quelltextanalyse (Syntaxanalyse während der Programmierung), Syntaxhervorhebung, das Springen im Quelltext (gehe zu Definition, Basistyp, Verwendungen, etc) oder die Parameterinfo-Anzeige von Methodensignaturen (IntelliSense).

Für Bibliotheken kann unterdessen die Entwicklungsumgebung frei in Bezug auf die verwendete Technologie gewählt werden. Dazu stehen neben kostenpflichtigen Umgebungen (e.g. Visual Studio .NET 200x) auch ausgereifte kostenfreie Varianten zur Verfügung (Eclipse Java, Eclipse CDT C/C++, Sharp Development).

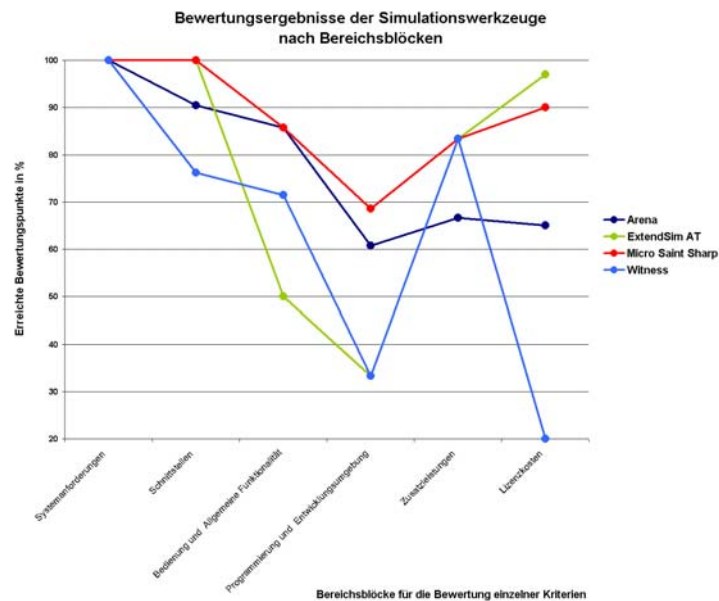
**Unterstützung von Techniken zur Verifikation und Validierung:** Die näher untersuchten Simulationswerkzeuge zeigten, dass bisher keine Unterstützung zur automatisierten Verifizierung und Validierung von Simulationsmodellen existieren.

Unter Verwendung von Simulationsbibliotheken würde sich diesbezüglich jedoch die Verwendung von Unittests (Modultests) hervorragend anbieten, welche sich in der Softwareentwicklung als Standardtechnik für das Testen von Softwaremodulen etabliert hat. Dadurch könnten beispielsweise Festwerttests, Monitoring anhand statistischer mindest oder maximal Bedingungen, Grenzwerttests mit Extremwerten als Eingabedaten oder die Prüfung zu erreichender Konfidenzintervalle oder eine Traceanalyse automatisiert und durchgeführt werden. Auch für Simulationswerkzeuge mit einer entsprechenden Schnittstelle könnte solch eine Unittest-Methodik implementiert werden.

## 5 Fazit

Im Folgenden werden die Ergebnisse der Analyse kurz zusammengefasst. Für eine ausführliche Erläuterung sei auf [3] verwiesen:

**Bewertungsergebnisse der Simulationswerkzeuge:** Die genannten vorselektierten Simulationswerkzeuge wurden ausgiebig aufgrund der erhaltenen Versionen der Hersteller auf dessen Funktionalität überprüft. Dabei wurde die Benutzeroberfläche hinsichtlich der grafischen und programmatischen Modellierung und Debugmöglichkeiten getestet, die Dokumentation auf dessen Inhalt überprüft, gezielt nach diversen Funktionalitätsbeschreibungen gesucht und verschiedene Beispielmodelle getestet. Sonstige Funktionalitäten wurden ebenso aufgrund der festgelegten Bewertungskriterien überprüft und bewertet. Folgende Abbildung zeigt die Resultate der Analyse und Bewertung.



**Bild: 3** Blockbewertung Simulationswerkzeuge, Quelle [3]

Die Analyse und Bewertung der einzelnen Werkzeuge hat schließlich gezeigt, dass alle Werkzeuge mind. einen *Drag n Drop*-ähnlichen Mechanismus für das Platzieren der Modellbausteine bzw. das Modellieren des Simulationsmodells verwenden. In diesen Bausteinen kann in hinterlegten Quelltextdateien oder in einzelnen Eingabebereichen (Quelltext) die spezielle Logik des Modells implementiert werden. Des Weiteren wurde ersichtlich, dass einige Werkzeuge skriptsprachenähnliche Programmiersprachen verwenden (Interpretersprachen) und dadurch oft keine Kompilierungsschritte vor der Ausführung des Modells benötigen. Diese sind jedoch nur bedingt für umfangreiche und komplexe Simulationsmodelle geeignet, da diese meist nur eingeschränkte Datentypen (keine dynamischen Listen, Vektoren, Hashtabellen, etc.), und vereinfachte Modellierungsmöglichkeiten (keine eigene Objekte, Vererbung, Datenkapselung, etc.) bieten.

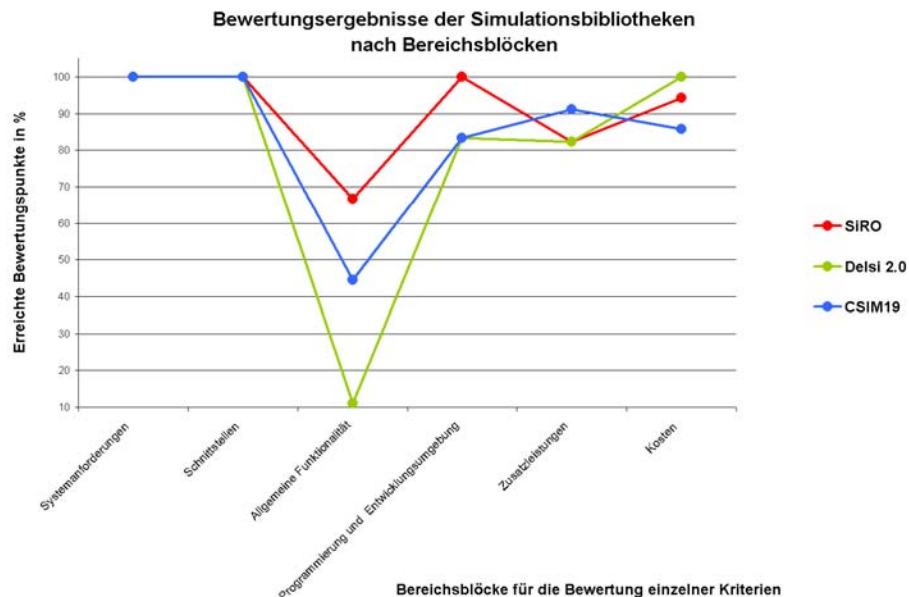
Hinsichtlich der Entwicklungsumgebungen wurde ersichtlich, dass deren Debugfunktionalität oder Optionen für das Implementieren zusätzlicher Logik nicht den Funktionalitäten aktuell verfügbarer Entwicklungsumgebungen entsprechen. Die integrierten Entwicklungsumgebungen der Werkzeuge sind in den meisten Fällen nur als befriedigend einzustufen und bieten nicht die gewünschte Entwicklungsunterstützung und Komfort, wie es beispielsweise bei den aktuellen Entwicklungswerkzeugen (Microsoft Visual Studio 2008, SharpDevelop, XCode oder bei Eclipse-basierenden Werkzeugen) üblich ist.

Die Kriterien im Bereich *Programmierung und Entwicklungsumgebung* haben sich für die bewerteten Simulationswerkzeuge als sehr anspruchsvoll herausgestellt, da hier alle Werkzeuge unzureichende Ergebnisse erzielten. Die Kriterien sind jedoch von großer Bedeutung, da die bisher entwickelten Simulationsmodelle für AWPS einen hohen Anteil an Quelltext für die modellgenerierende Logik beinhalteten und somit auch die

Programmierung einen großen Anteil bei der Erstellung eines Simulationsmodells einnimmt.

Weiters wurden die Schnittstellen der einzelnen Werkzeuge untersucht und es stellte sich heraus, dass diese zwar eine Steuerung von Außen bzw. von externen Programmen (bspw. AWPS Umgebung) zulassen, dadurch aber wesentliche Nachteile entstehen (unterschiedliche Technologien, verteilte Modelllogik, keine technologieübergreifende Fehlersuche/Debugging) und komplexe Kompilierungsschritte vor jedem Simulationslauf voraussetzen.

**Bewertungsergebnisse der Simulationsbibliotheken:** Die vorselektierten Bibliotheken wurden ebenfalls, wie die Werkzeuge, anhand der zur Verfügung gestellten Versionen, Dokumentation und der Beispielmodelle untersucht. Da sich die Bibliotheken grundsätzlich von den Werkzeugen unterscheiden, wurden diverse Bewertungsblöcke und Kriterien angepasst. Aufgrund der festgelegten Bewertungskriterien wurden die Bibliotheken evaluiert (siehe Bild 5).



**Bild: 4** Blockbewertung Simulationsbibliotheken, Quelle [3]

Bei den drei genannten Bibliotheken wird hinsichtlich der Modellierung das Simulationsmodell durch Quelltext realisiert. Die Bibliotheken bieten durch ihre objektorientierten Programmiersprachen (C++ und .NET C#) und frei definierbarer Logikverteilung, im Gegensatz zu den Simulationswerkzeugen, eine maximale Modellierungsflexibilität.

Zudem mussten für die Bibliotheken keine Entwicklungsumgebungen analysiert werden, da diese nicht Gegenstand der Bibliotheken sind und frei gewählt werden können. Durch den Einsatz einer Bibliothek kann der Anwender die bisher bestehende Entwicklungsumgebung weiter verwenden (keine Einarbeitungszeit).

Durch die Analyse wurde weiters ersichtlich, dass die Bibliotheken hinsichtlich Systemanforderungen und Schnittstellen beste Eigenschaften liefern und die Kriterien im vollen Umfang erfüllen. Auch hinsichtlich der Zusatzleistungen, welche das Schulungsangebot, Supportqualität und die kontinuierliche Weiterentwicklung betreffen, wurden durchwegs gute Werte erreicht.

Wie ebenfalls abgebildet, konnten die Bibliotheken jedoch dem Kriterienbereich *Allgemeine Funktionalität* nicht gerecht werden. Dieser Bereich bezieht sich auf die bereits vorhandene Funktionalität für Produktions- und Logistik-Simulationen, die Funktionalitätserweiterung und der Testqualität der Bibliothek. Die Funktionalität für Produktions- und Logistik-Simulationen beschreibt dabei etwa das Vorhandensein einer Vielzahl von vordefinierten Modellbausteinen. In diesem Bereich konnte sich die Bibliothek SiRO am besten durchsetzen, da diese speziell für Produktions-Simulationen entwickelt wurde und daher auch einige spezielle Konzepte und Modellkomponenten für diesen Bereich mitliefert.

Abschließend wurde ersichtlich, dass die analysierten Bibliotheken sich in den Bewertungsergebnissen nicht wesentlich voneinander unterscheiden und den Anforderungen in den meisten Fällen gerecht werden konnten.

## 6 Literatur

- [1] *Banks, J, Carson II John S und Nelson Barry L* : Discrete-Event System Simulation, 4. Aufl. New Jersey: Prentice Hall, 2005, S. 96-102.
- [2] *Law, A, M* : Simulation Modeling and Analysis, 4. Auflage, New York: McGraw-Hill, 2007, S. 195.
- [3] *Speckle, M* : Evaluierung und Analyse von Simulations Engines für die Simulation von Logistik- und Produktionsprozessen mit der Anwendungsplattform Simulation, Dornbirn, 2009.
- [4] *Wenzel, S* : Modellbildung und Simulation in Produktion und Logistik - Stand und Perspektiven. Dresden, ASIM Workshop, 2009, S. 7.
- [5] *März, L; Saler, M* : Ein Analyse-, Planungs- und Entscheidungsinstrument für Lagerlogistikanwendungen, Berlin, 13. ASIM Fachtagung, 2008, S. 237-245
- [6] *März, L* : Anwendungsumgebung zur Simulation und Optimierung von Transportnetzen, Wiesbaden, Praktische Anwendung der Simulation im Materialflussmanagement, 2008, S. 106-119
- [7] Operations Research and the Management Sciences (2007), 'Simulation software survey', <http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation.html>  
Letzter Zugriff: 05.06.2009.
- [8] *Banks, J., Carson, J., Nelson, B. L. & Nicol, D.* :Discrete-Event System Simulation, 4. Auflage, Prentice Hall, New Jersey 2004.

- [9] *Rizzoli, A.* : A collection of modelling and simulation resources on the internet',  
<http://www.apto101.com.br/projetofinal/referencias/SimulationTools.html>  
Letzter Zugriff: 05.06.2009.
- [10] Sindh2UK, M. Y. J.: Modelling & simulation resources',  
[http://homepage.ntlworld.com/myjamro/research/already\\_in\\_research/simulation\\_tools.htm](http://homepage.ntlworld.com/myjamro/research/already_in_research/simulation_tools.htm)  
Letzter Zugriff 08.06.2009.