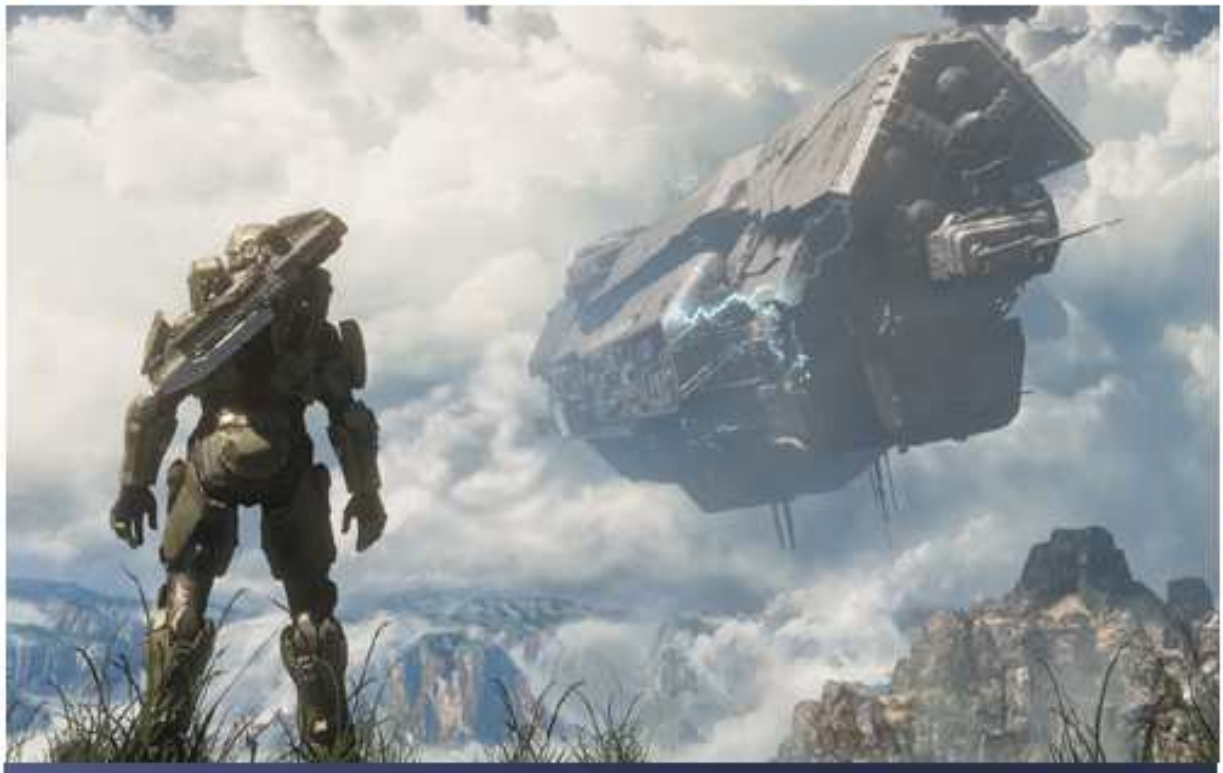
Zürcher Hochschule
für Angewandte Wissenschaften

Distributed Real Time Systems



Modul Systemarchitektur - Software Architektur
30.11.2015
Daniel Liebhart

Software Architektur – Distributed Real-Time Systems

Daniel Liebhart, 30.11.2015

Verfasser: Daniel Liebhart
Lektorat: Erika Paneth (2. Auflage)
10. Auflage: 2015
Version 10.0

Inhalt

Referenzen und Abkürzungen

Referenzen

[Bricken, Coco 1994]	W. Bricken & G. Coco: The VEOS Project - Presence-Teleoperators and Virtual Environments, Human Interface Technology Laboratory University of Washington, 1994
[Caltagirone et al. 2002]	S. Caltagirone, M. Keys, B. Schlief, M.J. Willshire: Architecture for a Massively Multiplayer Online Role Playing Game Engine, University of Portland 2002
[Dahmann et al. 1997]	J.S. Dahmann, R.M. Fujimoto, R.M. Weatherly: The Departement of Defense High Level Architecture, Proceeding of the 1997 Winter Simulation Conference, 1997
[Dodsworth 1997]	C. Dodsworth: Digilal Illusion: Entertaining the Future with High Technology, Addison-Wesley, August 15, 1997
[Gregory, Lander 2009]	J. Gregory, J. Lander: Game Engine Architecture, Taylor & Francis Ltd. 2009
[Kazman 1993]	R. Kazman: HIDRA: An Architecture for Highly Dynamic Physically Based Multi-Agent Simulations, International Journal for Computer Simulation, 1993
[Macedonia et al. 1997]	M.R. Macedonia, M.J. Zyda: A Taxonomy for Networked Virtual Environments (IEEE Multimedia, Jan-Mar 1997)
[Ohshima et al. 1998]	T. Ohshima, K. Satoh, H. Yamamoto, H. Tamura: AR2 Hockey: A Case Study of Collaborative Augmented Reality, IEEE Virtual Reality Annual International Symposium, 1998
[Smed et al. 2002]	J. Smed, T. Kaukoranta, H. Hakonen: A Review on Networking and Multiplayer Computer Games. Turku Centre of Computer Science Technical Report No 454 April 2002
[Steuer 1992]	J. Steuer: Defining Virtual Reality: Dimensions Determining Telepresence, Department of Communication, Stanford University, Journal of Communication 1992
[Strassburger 2000]	S. Strassburger: Distributed Simulation Based on the Level Architecture in Civilian Application Domains, Dissertation, Magdeburg 31.10.2000
[Yi-Bing et al. 1999]	L. Yi-Bing, P. Bellcore, A.F. Paul: Parallel Discrete Event Simulation, University of Florida, 1999

Abkürzungen

AI	Artificial Intelligence
CORBA	Common Object Request Broker Architecture
DIS	Distributed Interactive Simulators
FOM	Federation Objekt Model
HIDRA	Highly Interactive Distributed Real-Time Architecture
HLA	High Level Architecture
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IFSpec	HLA Interface Specification
MCG	Multiplayer Computer Games
MLFQ	Multilevel Feedback Queue
NVE	Networked Virtual Environments
OMT	Object Model Template
PDES	Parallel Discrete Event Simulation

PDU	Protocol Data Unit
RTI	Runtime Infrastructure
SIMNET	Simulation Network
SOM	Simulation Object Model
TBV	Temporal Bounding Volumes
UFR	Update Free Regions
VR	Virtual Reality

1 Einleitung

1.1 Distributed Real-Time Applications

Distributed Real-Time Applications existieren seit über 20 Jahren. Diese Klasse von Applikationen umfassen vor allem Simulationen, virtuelle Umgebungen und Computerspiele und stellen an die Architektur höchste Ansprüche, da es sich um eine logische Nachbildung der menschlichen Umgebung handelt.

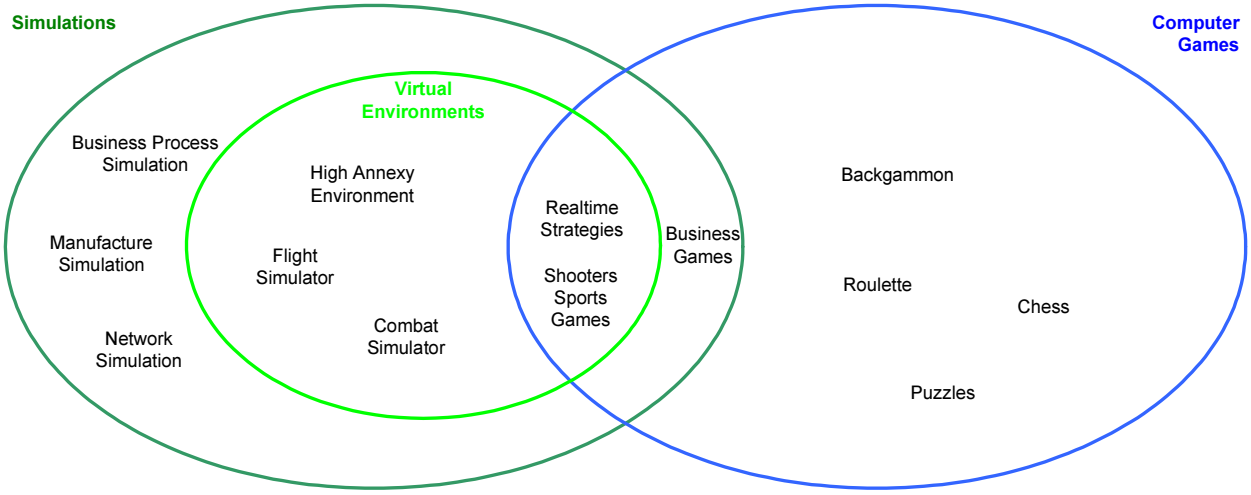


Abbildung 1: Distributed Real-Time Applications - Beispiele

Distributed Real-Time Applications lassen sich in drei Gruppen aufteilen:

- **Simulationen:** Die Nachbildung komplexer Vorgänge unter Einbezug einer Vielzahl von Usern.
- **Networked Virtual Environments:** Virtuelle Räume, in denen Menschen gemeinsam arbeiten, obwohl sie sich an völlig verschiedenen Orten aufhalten.
- **Multiplayer Computer Games:** Interaktive Spiele mit mehreren Teilnehmer / Teilnehmerinnen.

Distributed Real-Time Applications werden auch "Shared Space Technologies" genannt.

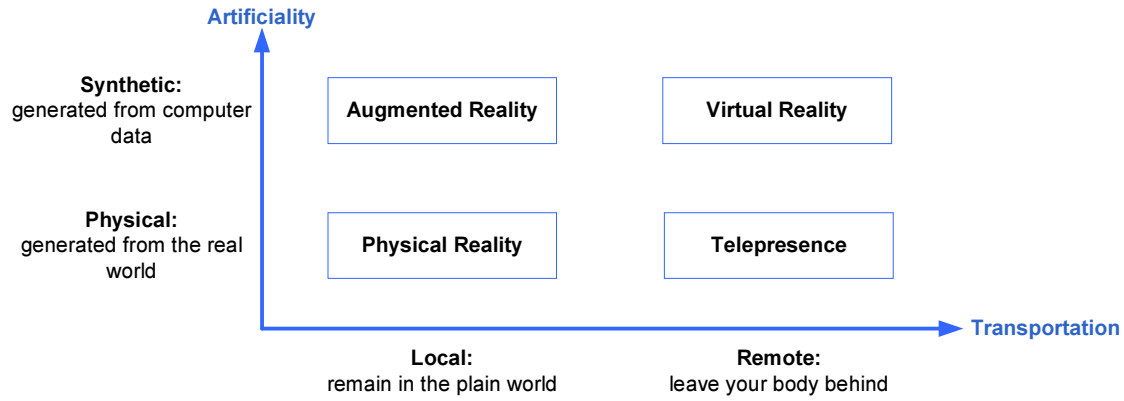


Abbildung 2: Klassifikation von "Shared Space Technologies" [Smed et al. 2002]

- **Augmented Reality:** Unter erweiterter Realität versteht man die visuelle Überlagerung von virtueller Information mit der Realität in Echtzeit. Dabei wird die Information am richtigen geometrischen Ort dargestellt werden.
- **Virtual Reality:** Vom Computer berechnetes, dreidimensionales Bild eines in der Realität tatsächlich vorhandenen oder auch fiktiven Raumes mit der Möglichkeit, sich frei in diesem Raum zu bewegen und interaktiv zu handeln. (Weitere Definitionen im folgenden Kapitel).
- **Physical Reality:** Die reale Welt soweit wir sie begreifen.
- **Telepresence:** Durch Verwendung von Technologie erscheint ein Individuum lokal präsent, obwohl es sich physisch an einem anderen Ort befindet.

1.2 Definitionen der Virtuellen Realität

Der Begriff Virtual Reality ist von vielen Wissenschaftlern definiert worden. Das Konzept selbst wird und der Begriff Cyberspace wird in der Neuroromancer Trilogy von William Gibson ausführlich beschrieben und definiert:

> **'Think of Jackie as a deck, Bobby, a cyberspace deck, a very pretty one with nice ankles […] Think of Danbala, who some people call the snake, as a program. Say as an icebreaker. Danbala slots into the Jackie deck. Jackie cuts ice. That's all.'**
> **'Okay,' Bobby said, getting the hang of it, 'then what's the matrix? If she's a deck, and Danbala's a program, what's cyberspace?'**
> **'The world,' Lucas said.**

Andere Definitionen ([Steuer 1992]):

> **Virtuelle Realität ist eine elektronische Simulation einer Umgebung, die einem Nutzer / einer Nutzerin eine Interaktion in einer realistischen 3D Umgebung durch geeignete Interface-Devices (Head-Mounted Display, Wired Clothes) erlaubt.**

> **Die Virtuelle Realität ist eine alternative Welt, die mittels computergenerierten Bildern gestaltet wird und auf menschliche Bewegungen reagiert. Diese simulierten Umgebungen werden normalerweise mittels einem Data Suit besucht.**

> **Die Begriffe Virtuelle Welt, Virtuelle Cockpits und Virtuelle Workstations werden als Beschreibung bestimmter Projekte verwendet… Bereits 1998 hat Jaron Lanier, der CEO von VPL, den Begriff verwendet, um all diese Projekte unter einer separaten Rubrik laufen zu lassen. Der Begriff bezieht sich also typischerweise auf drei dimensionale Realitäten.**

> **Der Schlüssel zur Definition der Virtuellen Realität als menschliches Erlebnis ist die Präsenz. Präsenz kann als Erfahrung in einer physischen Umgebung gedacht werden; Es bezieht sich nicht auf**

die Umgebung einer physischen Welt, sondern auf die Wahrnehmung dieser Umgebung, die sowohl durch automatische als auch durch kontrollierte Mentale Prozesse geschieht.

1.3 Allgemeine Architekturkriterien und Distributed Real-Time Applications

Kriterium	Impact Distributed Real-Time Applications
Performance	Reaktionszeit des Systems ist zwingend in Echtzeit.
Security	Schutz des Individuums und der verwendeten Daten ist in einer verteilten Umgebung besonders kritisch.
Scalability	Die ständig wechselnde Anzahl von User darf keinen Einfluss auf die System Performance haben.
Availability	Das System muss 7x24 h verfügbar sein.
Usability	Die Interaktion in einem 3D Environment erfordert spezielle Peripheriegeräte (Controller, Head-Displays, Data-Gloves, 3D Maus, Motion Capture Cameras, …).
Flexibility	Das System muss dynamisch neue virtuelle Umgebungen verwalten koennen.
Portability	Mehrere Geräte und Infrastrukturkomponenten müssen unterstützt werden.
Reusability	Die Replizierbarkeit verschiedener Schichten und Komponenten des Systems muss gegeben sein.
Testability	Das Testen einer solchen Umgebung erfordert die Realisierung spezieller Testgeräte und Testmaschinen.
Separation of Concern	Die einzelnen Komponenten des Systems müssen klar getrennt warden, da sich jede Vermischung von funktionalen Einheiten sofort auf die Gesamtperformance des Systems auswirkt.
Comprehension	Das Management einer Real-Time Umgebung erfordert eine hohe Verständlichkeit der Aufgabe der einzelnen Komponenten.
Correctness / Completeness	Das System muss mit Engpässen umgehen können und den Verlust ganzer System- und Infrastrukturteile ausgleichen können.
Referencial Transparency	Die hohe Granularität und die Verteilung der einzelnen Komponenten erfordert die Abstraktion der Funktionsweise durch Interfaces.
Buildability	Die Endgeräte des Systems müssen einfach und billig sein.
Coupling	Es ist in jedem Fall eine "loose Coupling" zu realisieren, da die einzelnen Komponenten möglichst schmale Schnittstellen und möglichst wenig Datenaustausch aufweisen müssen.
Cohesion	Die Cohesion wird den Anforderungen des Gesamtsystems untergeordnet.

1.4 Zentrale Faktoren

1.4.1 Hardware der Endgeräte

Neben dem Standard-PC's sind Spiel-Konsolen und zunehmend auch Mobile Devices die am meisten verwendeten Endgeräte im Bereich Distributed Real-Time Applications. Seltener werden Data-Gloves, Data-Suits und Caves verwendet. Die neueste Generation dieser Systeme verwendet zunehmend so genannte Motion Capture Cameras, um die Bewegungen eines Menschen aufzunehmen und so eine direkte Interaktion zu erlauben.

1.4.2 Beschaffenheit des Netzwerkes

Die Beschaffenheit des Netzwerkes ist der zentrale Faktor für die Interaktivität einer Distributed Real-Time Application.

Aus Architektursicht sind zwei logische Faktoren entscheidend:

- **Communication:** Die Beschaffenheit der logischen Verbidungen zwischen den Knoten eines Netzwerkes (Peer To Peer versus Client Server sowie seltene Multicast Networks)
- **Data & Control:** Die Art und Weise wie Informationen in einem Netzwerk gespeichert und verwaltet werden (zentralised versus replication).

Die physikalischen Eigenschaft eines Netzwerkes wird durch die Beschaffenheit der einzelnen Devices (Router, Switches, Nodes) sowie die Bandbreite bestimmt.

1.5 Hintergrund

Das erste System im Bereich Simulation wurde 1976 von Jack Thorpe, einem Wissenschaftler der Air Force Base Phoenix, Arizona USA, entworfen. Es diente dem Gruppentraining von Piloten. Entwickelt wurde das System basierend auf Standalone Flugsimulatoren. 1983 wurde von Thorpe und seinem Team in der Advanced Research Projects Agency die Idee zu SIMNET (SIMulator NETworking project) entwickelt. Das Projekt wurde teilweise ausgeschrieben und der Firma Bolt Beranek & Newman Cambridge, Massachusetts wurde der Auftrag erteilt, die Netzwerk- und Simulations-Software zu entwerfen. Ducan Miller der Projektleiter der SIMNET Lösung war Mitglied der Society of American Magicians. Basierend auf den Erkenntnissen der Zauberer über das Verhalten und die Aufmerksamkeit von Menschen wurden die Grundeinheiten von SIMNET definiert.

SIMNET basiert auf einer gemeinsamen Welt, künstlichen Einheiten (Avatare), die Objekte mit definierten Eigenschaften darstellten und Simulatoren, die die Aufgabe hatten, einen Teil der Welt sowie die Avatare zu beobachten.

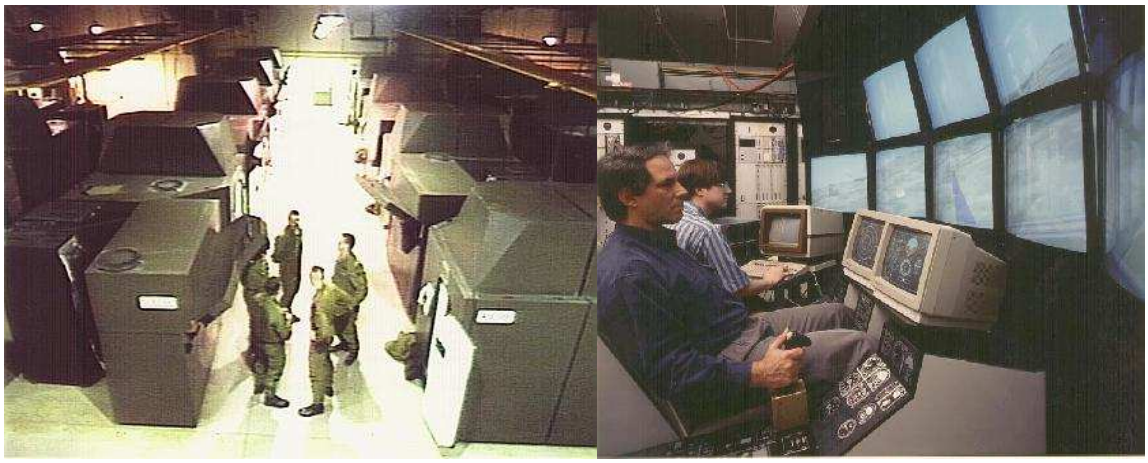


Abbildung 3: SIMNET Simulatoren in Ford Knox [Dodsworth 1997]

1985 wurde die erste Version erstellt, 1990 umfasste das System 238 Simulatoren, die jeweils einen Teilaspekt eines Kampfgeschehens darstellten, wie beispielsweise einen Panzer. Die Darstellung wurde von CIG (Computer Image Generation) Systemen übernommen, die auf einzelnen Rechnern abliefen. Im Gegensatz zu bestehenden Flugsimulatoren, die jeweils auf Grossrechnern alle 5 CIG (das Minimum für einen Flugsimulator) ausführten, waren diese Recheneinheiten billiger, sie kosteten weniger als 100'000 $ pro CIG. Das Ziel von SIMNET war ein Kampftraining basierend auf realen Daten. Zu diesem Zweck wurde die Schlacht „Battle of 73 Easting“, die am 26. Februar 1991 im Golfkrieg stattgefunden hat und genau drei Tage dauerte, vollständig in der Simulation nachgebaut. Die Umsetzung erfolgte einen Monat nach der Schlacht basierend auf einer umfangreichen Datensammlung (Action Logs, Radio und anderen elektronischen Informationen) sowie auf Interviews einer Vielzahl beteiligter Soldaten. Innerhalb von 9 Monaten wurde die Simulation erstellt.

SIMNET war das erste voll einsatzfähige Virtual Reality System überhaupt.

Heute nennt man diese Systeme Distributed Interactive Simulators (DIS) und sie unterstützen bis 10'000 menschliche Spieler sowie 9000 Software Avatare.

2 High Level Architecture

2.1 Einleitung

Die High Level Architecture (HLA) ist eine generelle Architektur für Simulationssysteme [Dahmann 1997]. Sie wurde 1996 vom DoD basierend aus den Erfahrungen mit SIMNET definiert und 2000 als IEEE Open Standard 1516 spezifiziert.
Diese so gennante Baseline Definition umfasst:

- **HLA Rules:** Die HLA Rules umfassen 10 Regeln, die definieren, wie sich ein HLA System verhält.
- **HLA Interface Specification (IFSpec):** Die Schnittstelle zwischen den HLA Federates (Simulations, Support Utilities und Interfaces to life players) und der HLA Runtime Infrastructure.
- **HLA Object Model Template (OMT):** Eine Vorlage für die Spezifikation von HLA Objektmodellen.

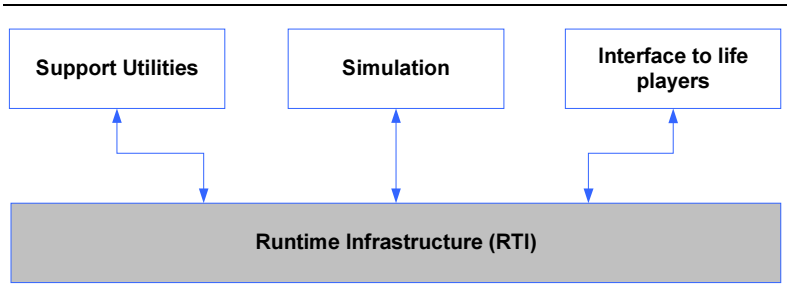


Abbildung 4: High Level Architecture

HLA definiert eine Art Protokoll, welches die Datenübertragung während einer Simulation kontrolliert und steuert. Das Herzstück dieses Protokolls ist die Laufzeitumgebung (Runtime Infrastructure – RTI), welch Basis-, Koordinations- und Kommunikationsdienste zur Laufzeit bereitstellen. Die Federates (Teilnehmer einer gemeinsamen Simulation) interagieren über die RTI. Basierend auf Objektmodellen kommunizieren Federates und RTI miteinander, ein direkter Datenaustausch zwischen Federates findet nicht statt.

2.2 Federation

Definition einer HLA-Federation (Verbund, Föderation):

> **Eine Federation ist eine Sammlung von Federates, die über RTI interagierend, einen Simulationlauf (Federation Execution) durchführen.**

Die Durchführung einer Federation Execution erfolgt mittels Austausch der Objektmodelle (SOM: Simulation Object Model) zwischen einzelnen Federates über die RTI.

2.3 HLA Rules

Federation Rules:

- Federations müssen mit Hilfe des Object Model Template dokumentiert werden, also ein High Level Architecture Object Model besitzen.
- Die Instanzierung von Simulation Object Model Objekten findet nur in Federates, nicht in der Runtime Infrastructure statt.
- Der Datenaustausch zwischen Objektinstanzen in unterschiedlichen Federates erfolgt durch die Runtime Infrastructure.
- Ein Attribut einer Objektinstanz darf nur einem Federate zugeordnet sein.

Federate Rules:

- Federates werden durch ein SOM (Simulation Object Model) beschrieben. Die Spezifikation erfolgt gemäss OMT.
- Federates publizieren Objektattribute in ihrem SOM, können die Attribute jedoch auch wieder aus dem Modell entfernen. Messages werden zwischen einzelnen SOM Objektinstanzen ausgetauscht.
- Die Bedingungen für Aktualisierungen von Objektattributen sind veränderbar.
- Federates übernehmen die zeitliche Koordination mit anderen Federates für den Datenaustausch.

2.4 HLA Interfaces

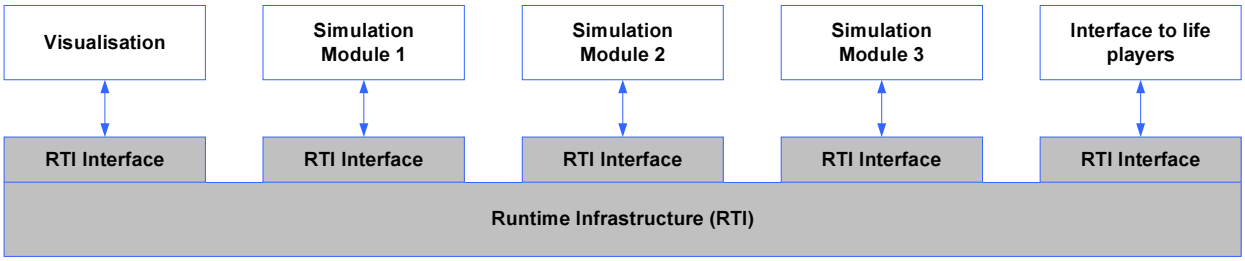


Abbildung 5: HLA Schnittstellen

Das HLA Interface besitzt folgende Eigenschaften:

- Die Schnittstelle erlaubt das Schreiben und Lesen von Modellvariablen.
- Die Schnittstelle steuert den Zeitraum zwischen einzelnen Simulationsintervallen.
- Die Schnittstelle erlaubt die Begrenzung des nächsten Simulationsinterwalles sowie die Ermittlung des nächsten Intervalls und die Wiederholung des letzten Simulationsinterwalles.
- Das HLA Interface ist die einzige Interaktionsmöglichkeit zwischen verschiedenen Komponenten eines Simulationssystems.
- Die Schnittstellen stehen als C++, JAVA, ADA API sowie als CORBA IDL zur Verfügung.

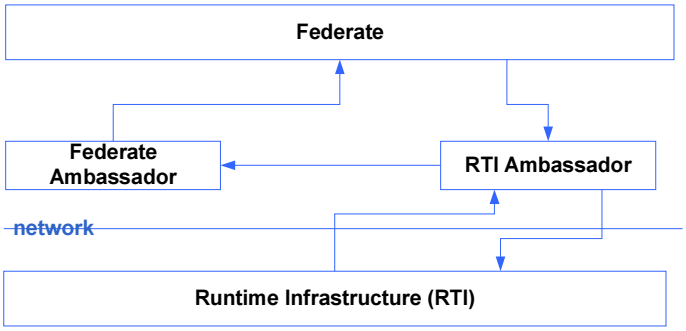


Abbildung 6: Federate – RTI Interface Ambassadors

Die Interaktion zwischen allen Komponenten (Federates) eines auf der HLA basierenden Systems erfolgt über "Ambassadors". Die Basisdienste der Schnittstellen werden über diese Ambassadors implementiert. Ein Federate kommuniziert mit dem RTI über dessen Ambassador. Ein Ambassador realisiert die Objekte und die Kommunikation eines Federates.

2.5 Runtime Infrastructure (RTI)

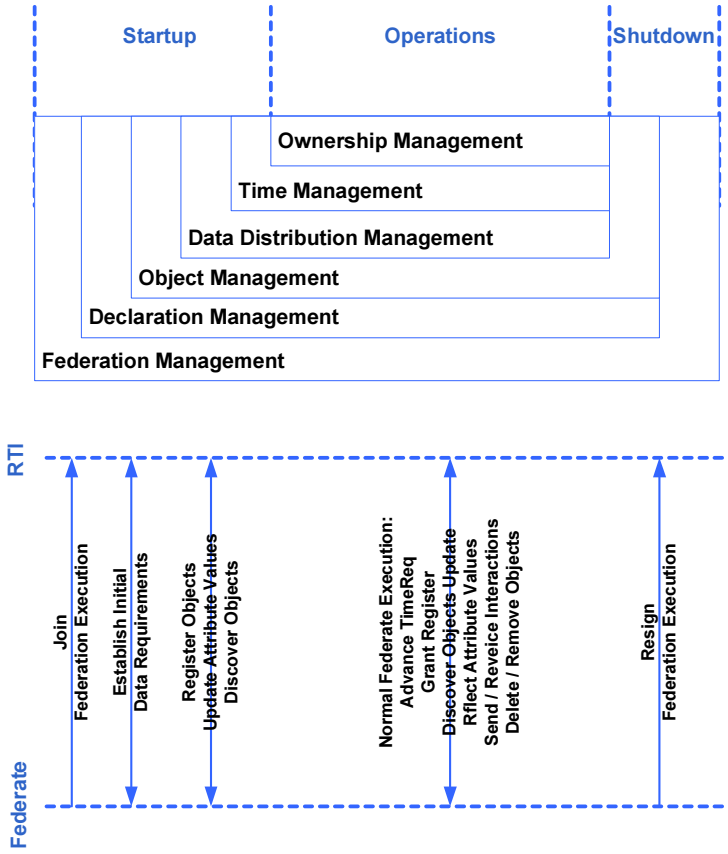


Abbildung 7: RTI Funktionen und Interaktion mit einem Federate [Strassburger 2000]

- **Federation Management:** Das Erzeugen, Join / Resign sowie das Anhalten und Speichern von Federates respektive Federation Execution.

- **Declaration Management:** Das Abonnieren und Publizieren von Informationen.
- **Object Management:** Die Manipulation von Objektinstanzen und der Nachrichtenaustausch zwischen Federates.
- **Data Distribution Management:** Die Datenübermittlung (in den meisten konkreten Implementationen über TCP/IP).
- **Time Management:** Die transparente Koordination der Simulationsschritte sowie die Interoperabilität über Zeitzonen hinweg.
- **Ownership Management:** Der Transfer der Rechte zur Änderung von Objektattributen.

2.6 Object Model Template (OMT)

Das Object Model Template dokumentiert eine Simulation. Die Simulationselemente werden alle auf Objektebene abgebildet und entsprechend realisiert. Das OMT beinhaltet:

- **Obect Class Structure Table:** Die Objektbeschreibungen der Federation und der Federates als hierarchische Struktur.
- **Object Interaction Table:** Die Ereignisse und Verhaltensweisen von Objekten und deren mögliche Attributmodifikation.
- **Attribute/Parameter Table:** Detaillierte Beschreibung der einzelnen Objekte und deren Events.
- **Complex Data Type Table:** Die Beschreibung der verwendeten Datentypen.
- **FOM/SOM Lexicon:** Das Verzeichnis aller an einer Simulation beteiligten Objekte.
- Das **Simulation Objekt Model (SOM)** beinhaltet alle Informationen deines Federates (Objekte, Attribute, Interaktionen, Events).
- Das **Federation Object Model (FOM)** stellt den Kontakt (auf Objektebene) zwischen einzelnen Federates dar und überprüft die Einhaltung der HTA-Rules.

2.7 Andere Simulations-Architekturen

2.7.1 Distributed Interactive Simulation

Distributed Interactive Simulation (DIS) ist der Vorgänger von HLA und ist im IEEE Standard 1278-1993 definiert.

IEEE Definition:

> **„DIS is a time and space coherent representation of world environments designed for linking the interactive, free play activities of people in operational exercises."**

Ziele:

- Festlegung der auszutauschenden Daten durch Definition von Protocol Data Units (PDU)
- Festlegung von Topologie und Schnittstellen
- Festlegung von Kopplungs- und Anwendungsregeln

Prinzip	Wirkung
Autonome Simulationsknoten	Vereinfacht Erstellung und Integration von Simulatoren in verteilte virtuelle Umgebungen. Jeder DIS Knoten ist verantwortlich für Zustände von einer oder mehreren Einheiten (z.B. ein Fahrzeug), sowie für die Darstellung des Zustands relevanter externer Einheiten (z.B. andere Fahrzeuge). Die Informationen werden über genormte Nachrichten (PDUs - Protocol Data Units) ausgetauscht. Die DIS Knoten bestimmen NICHT die Empfänger der Nachrichten. Die Zeitsteuerung erfolgt lokal (Local Clock).
Übertragung von tatsächlichen Zuständen	Jede Abweichung von Tatsacheninformationen muss beim Empfänger nachgeführt werden (Sichteinschränkungen sind zu modellieren)
Übertragung ausschließlich von Zustandsänderungen	Optimierung der Bandwidth-Usage: Nur Übertragung von Zustandsaktualisierungen.

Nutzung von Dead Reckoning Algorithmen	Um Anzahl von Zustandsaktualisierungen weiter zu reduzieren, werden Zustände lokal extrapoliert

2.7.2 Parallel Discrete Event Simulation

Parallel Discrete Event Simulation (PDES) ist ein Forschungsgebiet der Informatik, welches sich mit der effizienten Synchronisation von logischen Prozessen befasst [Yi-Bing et al. 1999]. Kernstück von PDES ist die Idee, dass Events abgearbeitet werden können, bevor sie angekommen sind ("unsave Events" – Optimistic Event Processing).

2.8 Zivile Simulations-Systeme

Name	Hersteller	Anwendung	Informationen
acslX	Aegis	Chemische Prozesse, Luftfahrt, Automobile, Landwirtschaftt	http://www.acslsim.com/
Arena	Rockwell Automation	Herstellungs- und Verpackungsprozesse, Geschäftsabläufe, Materialbewirtschaftung	http://www.arenasimulation.com/
Automod	SimPlan AG	Discrete Event Simulation für die Optimierung der Wirtschaftlichkeit eines Unternehmens, Materialflüsse, Gepächförderung, Montagelinien	http://www.automod.de/
ExtendSim	Imagine That Inc.	Engineering Design, wissenschaftliche Analysen	http://www.extendsim.com/index.html
SLX 2.0 (GPSS/H)	Wolverine Software	Herstellungsprozesse, Netzwerke, Gesundheitswesen	http://www.wolverinesoftware.com
ProModel, EPS	ProModel Corporation	Supply Chain, Kapazitätsanlayse, Herstellungsprozesse	http://www.promodel.com/
Simplex3	Uni Passau (Lehrstuhl für Operations Research und Systemtheorie)	Generelles Framework mit eigener Sprache	http://www.simplex3.net/
MATLAB / SIMULINK	The MathWorks, Inc.	Dynamische Simulationen, Kommunikationssysteme, DSP	http://www.mathworks.com/
Tecnomatix	Siemens	Fabriksimulation	http://www.plm.automation.siemens.com/en_us/products/tecnomatix/index.shtml

3 Networked Virtual Environments

3.1 Einleitung

Networked Virtual Environments (NVE) oder auch Distributed Virtual Environments sind virtuelle Umgebungen, in denen Anwenden miteinanden in Echtzeit interagieren.

Eigenschaften:

- Alle AnwenderInnen sind in demselben virtuellen Raum präsent.
- TeilnehmerInnen sind durch Avatare repräsentiert.
- Die Aktionen anderer TeilnehmerInnen werden in Echtzeit wahrgenommen.
- TeilnehmerInnen können miteinander kommunizieren.
- Es findet eine Interaktion mit anderen TeilnehmerInnen und virtuellen Objekten statt.

Die Anwendungen sind Games, Telemedizin, virtuelle Laboratorien, Simulationen. Networked Virtual Environments sind eine spezialisierte Art von Simulationen. Die grundsätzliche Architektur dieser Systeme unterscheidet sich nicht von der Architektur von Simulationssystemen.

3.2 Rahmenbedingungen

3.2.1 Zielsysteme

Nach verschiedenen Versuchen in den frühen 90er Jahren, ein generelles System für alle möglichen Arten von virtuellen interaktiven Umgebungen zu schaffen, hat sich herausgestellt, dass vor allem drei Bedingen die Beschaffenheit eines NVE bestimmen:

- Die Anzahl der TeilnehmerInnen, die sich in derselben virtuellen Welt aufhalten.
- Die Komplexität der Objekte und deren Verhaltensweisen
- Das Ausmass der Interaktion

3.2.2 Netzwerk

Die Netzwerktopologie und der Netzwerktyp bestimmt die Realisierung eines NVE. Die drei grundsätzlichen Typen der Netzwerkcharakteristika sind:

- **Connection:** Die Verbindung zwischen den einzelnen Infrastrukturkomponenten (Modem, Fixleitung, PeerTo-Peer).
- **Unicast:** Eine Meldung wird dediziert von einem bestimmten Sender an einen bestimmten Empfänger gesendet. Für ein Update eines Gesamtsystems sind O(N) Meldungen notwendig.
- **Multicast:** Eine Menge von Endgeräten kann mittels "Connectionless Messages" miteinander kommunizieren. Lediglich ein Update für alle beteiligten Entitäten ist notwendig.

3.2.3 Teilnehmende Rechner (Endgeräte)

Die Eigenschaften der teilnehmenden Rechner bestimmen sowohl die Beschaffenheit als auch die Grenzen eines NVE. NVE Systeme wie beispielsweise Teleconverencing benötigen keine leistungsstarken Rechner, während Hochauflösende Umgebungen mit vielen logischen Objekten bereits auf Ebene der Endgeräte sehr viel CPU und Memory erfordern.

3.2.4 Eingabegeräte

Eine Vielzahl von Eingabegeräte wird heute in NVE's verwendet. Von der Computermaus bis hin zum integrierten Datenanzug mit Helm. Die neuste Generation von Eingabegeräten kommt ohne irgend ein Zusatz aus, wie beispielsweise das Projekt Natal, welches von der Microsoft an der E3 Konferenz in Los Angeles 2009 vorgestellt hat. Es besteht aus einer an die Xbox 360 angeschlossenen Kamera, die Bewegungen und Stimmen erfasst. Das System besteht aus einer Kamera, einen Tiefensensor und einem Raumklangmikrophon und kann Körperbewegungen in 3D aufnehmen und gleichzeitig auf Befehle, Richtungsänderungen und Veränderungen der Stimmlage reagieren. Es wird heute als Kinect Erweiterung zur Xbox vertreiben.

3.3 Toolkits und Integrierte Architekturen

3.3.1 Beispiel: Architektur einer Game Engine

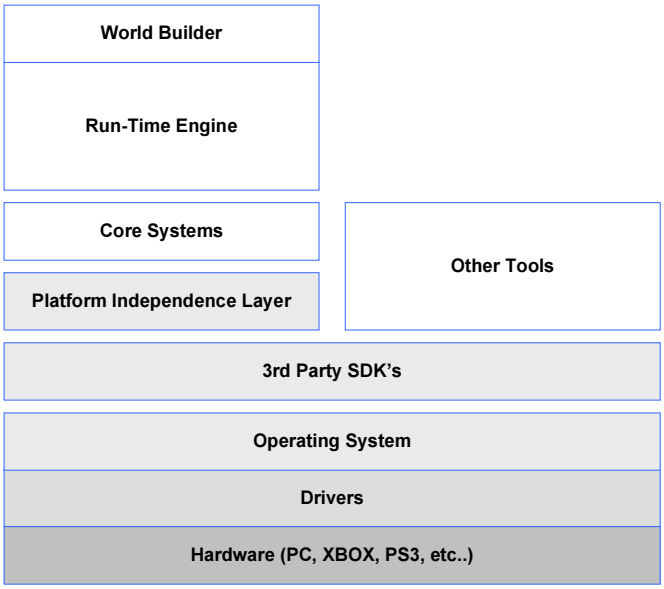


Abbildung 8: UnrealEngine Tool Architektur [Gregory, Lander 2009]

Jede Game Engine besteht aus einem Toolset, welches sowohl die Entwicklung des Games als auch das Game selbst zur Laufzeit unterstützt. Ein typisches Beispiel ist der Aufbau der UnrealEngine, dessen wichtigste Bestandteile, das Platform Independence Layer, die Core Systems, die Run-Time Engine und der World Builder sind.

- **Platform Independence Layer:** Dieses Layer ist die Voraussetzung dafür, dass Games auf verschiedenen Konsolen und anderer Hardware lauffähig sind. Es werden Operating System Calls, Datentypen, File System, Netzwerk, Timing und Graphic so dargestellt, dass ein einheitliches Verhalten der Anwendung über die verschiedenen Plattformen hinweg gewährleistet werden kann.
- **Core Systems:** Die Core Systems umfassen eine Reihe von Werkzeugen, wie Beispielsweise Memory Management, eine Mathematik-Library, Spezielle Datenstrukturen, die für jedes Game nützlich sind.
- **Run-Time Engine:** Die Run-Time Engine ist das Herzstück jedes Games und umfasst die Renderer, Animations- , Kollisiondetection- und Physics- und Visual-Effect-Generatoren sowie das Front-End, Audio und Interface Decive Processing und vieles anderes Mehr.

- **Wolrd Builder:** Jedes Game ist eine Multimedia Anwendung und besteht aus einer Vielzahl von Bildern, Video- und Audiosequenzen und anderen Elementen, die mit dem World Builder zu einer kohärenten Spielwelt zusammengesetzt werden.

3.3.2 Toolkits und Game Engines

Toolkits-basierte Architekturen umfassen Werkzeuge und Libraries zur Erstellung und Interaktion mit virtuellen Umgebungen. Sie umfassen unter anderem:

- Kontrolle der Objekte in der Umgebung
- Bewegung der Repräsentation des Avatars
- Dynamische Viewpoints
- Objekt-Relationship
- Display-Management
- Resourcen-Synchronisation

Name	Hersteller	Information
Doom rendering engine	ID Software	http://www.idsoftware.com/
Unreal Engine	Epic Games	http://www.unrealtechnology.com/
CryEngine	Crytek.	http://www.crytek.com/
Java3D	Oracle	http://www.java3d.org/
Bleder	Bleder Foundation	http://www.blender3d.org/
DI-Guy	Boston Dynamics	http://www.diguy.com/diguy/
Cortona	Parallel Graphics	http://www.parallelgraphics.com/products/

3.3.3 Integrierte Architekturen

Integrierte Architekturen liefern bereits ein voll funktionsfähiges System, welches alle Basisaufgaben eines NVE realisiert. Der Grundansatz für die Realisierung eines konkreten Systems besteht darin, die entsprechenden Komponenten zu ersetzen oder zu ergänzen.

Name	Hersteller	Information
VEOS	University of Washington	http://www.hitl.washington.edu/publications/r-93-3/
DIVE	Swedish Institute of CS	http://www.sics.se/dive/
MASSIVE (Lord of the Rings, Resident Evil)	Massive	http://www.massivesoftware.com/

3.4 Data- und Task-Distribution

In eine NVE ist die Data- und Task-Distribution niemals transparent. Die Verteilung ist die zentrale Aufgabe bei der Realisierung eines NVE. Verschiedene Ansätze zur Verteilung werden verwendet:

- **Sharing and Distributing the World:** Die gemeinsame virtuelle Welt wird auf die verschiedenen Rechner verteilt. Dabei wird die Welt aufgeteilt und die Zustandsänderungen in der Welt zwischen den bestehenden Rechnern aufgeteilt.
- **Sharing Behaviours:** Das Verhalten der Avatare und der Objekte der Welt wird auf die beteiligten Rechner aufgeteilt.

Vier Verteilungsschemen sind denkbar:

- **Separate Servers:** Aufteilung der Welt in unabhängige Welten.
- **Uniform Geometrical Structure:** Lineare Aufteilung der Welt.
- **Free Geometrical Structure:** Durch die User gesteuerte Verteilung der Welt.
- **User-Centrered Dynamic Structure:** Aufteilung der Welt aufgrund der Interaktion zwischen den einzelnen Usern.

3.5 Architekturen von Networked Virtual Environments

3.5.1 Shared Scalable Server

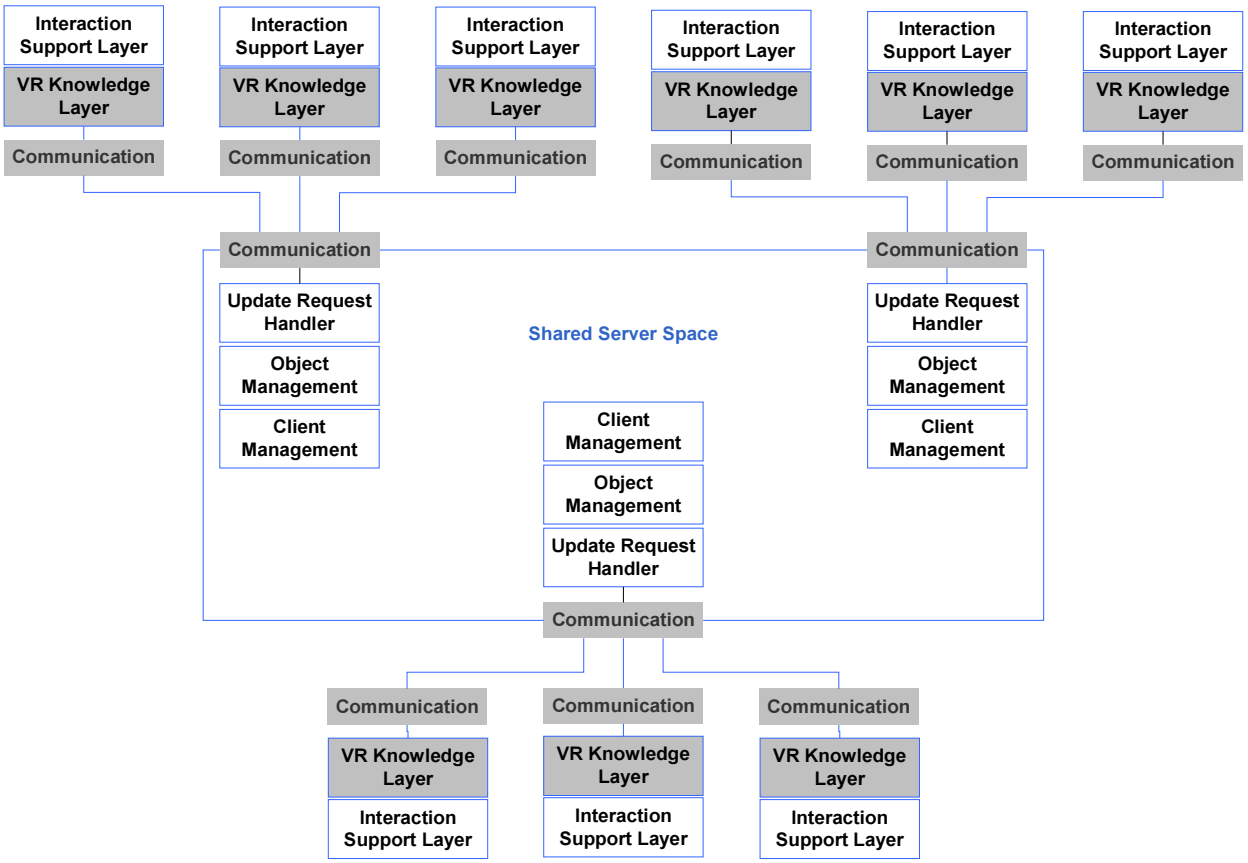


Abbildung 9: Beispiel VNE mit Shared Scalable Servers [Macedonia et al. 1997]

Die Lastverteilung der Server erfolgt dynamisch anhand der Anforderungen der Clients.

- **Interaction Support Layer:** User Interface der virtuellen Welt.
- **VR Knowledge Layer:** Die Verwaltung des Inhalts der virtuellen Welt, das heisst der einzelnen Objekte, die in den meisten Fällen vom Server gesendet worden sind. Lokale Objekte, die neu kreiert worden sind, werden an den Server über die Communication gesendet.
- **Communication:** Send / Receive Messages.
- **Client Management:** Verwaltung der clientspezifischen Informationen.
- **Object Management:** Verwaltung und Verfolgung der Zustände der Objekte, die von verschiedenen Clients platziert worden sind.
- **Update Request Handler:** Empfängt Updates der neuen Positionen von Objekten durch die Clients. Sendet die Updates an andere Clients.

3.5.2 Funktionale Architektur

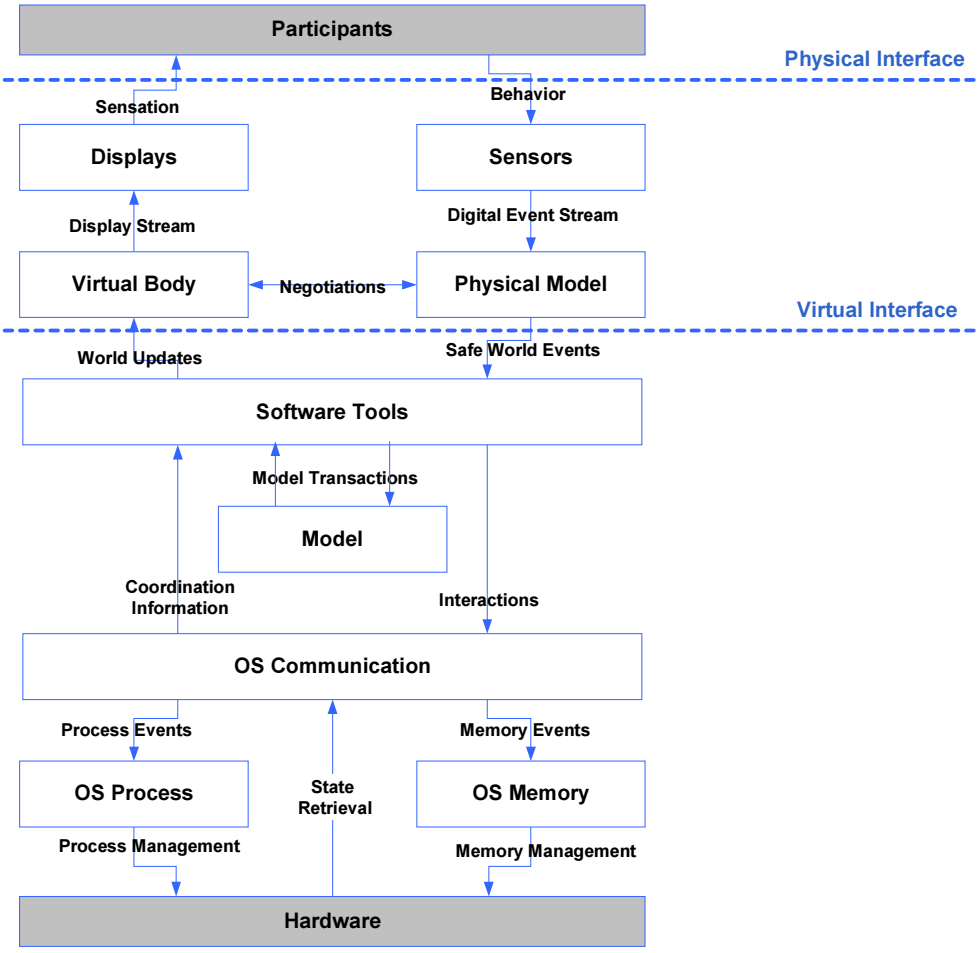


Abbildung 10: Funktionale Architektur eines Networked Virtual Environments [Bricken, Coco 1994**]**

Behavior & Sensory Transducing Subsystem:

- **Participant:** Systeme, die eine virtuelle Realität abbilden, sind dergestalt, dass sie einen User in das Gesamtsystem einbinden. Diese Person interpretiert die wahrgenommene Welt und agiert in derselben.
- **Displays (Output Devices):** Diese Komponente übersetzt die virtuelle Welt (als Display Stream) in eine Darstellung, die als Sinneseindruck durch den Menschen wahrgenommen wird.
- **Sensors (Input Devices):** Diese Komponente übersetzt das Verhalten des Users sowie Messungen der physikalischen Umwelt des Users in digitale Ereignisse.

Virtual Toolkit Subsystem:

- **Physical Model:** Das Physical Model ist für die Abbildung des digitalen Inputs in ein realistisches Model des Users und seiner Umgebung verantwortlich.
- **Virtual Body:** Diese Komponente passt die Effekte in einer virtuellen Welt (als Digital World Events) in die Darstellung eben dieser Welt ein.
- **Software Tools:** Sie dienen der Programmierung und Kontrolle der virtuellen Welt, stellen Navigationsmechanismen und andere Interatkionsmöglichkeiten zur Verfügung
- **Model:** Das Model der virtuellen Welt, oder auch die Welt genannt, ist die Datenbasis, welche sowohl die statischen als auch die dynamischen Attribute aller Objekte dieser Welt speichert.

Computional Subsystem:

Die Komponenten **OS Communication**, **OS Process**, **OS Memory** sowie die **Hardware** sind für die Abarbeitung der VR Software zuständig.

3.5.3 Highly Interactive Distributed Real-Time Architecture

Das System HIDRA (Highly Interactive Distributed Real-Time Architecture) wurde 1993 von der Carnegie Mellon University in Pittsburgh USA entworfen. Fokus des Entwurfs war die Realisierung einer generellen Softwarearchitektur [Kazman 1993].

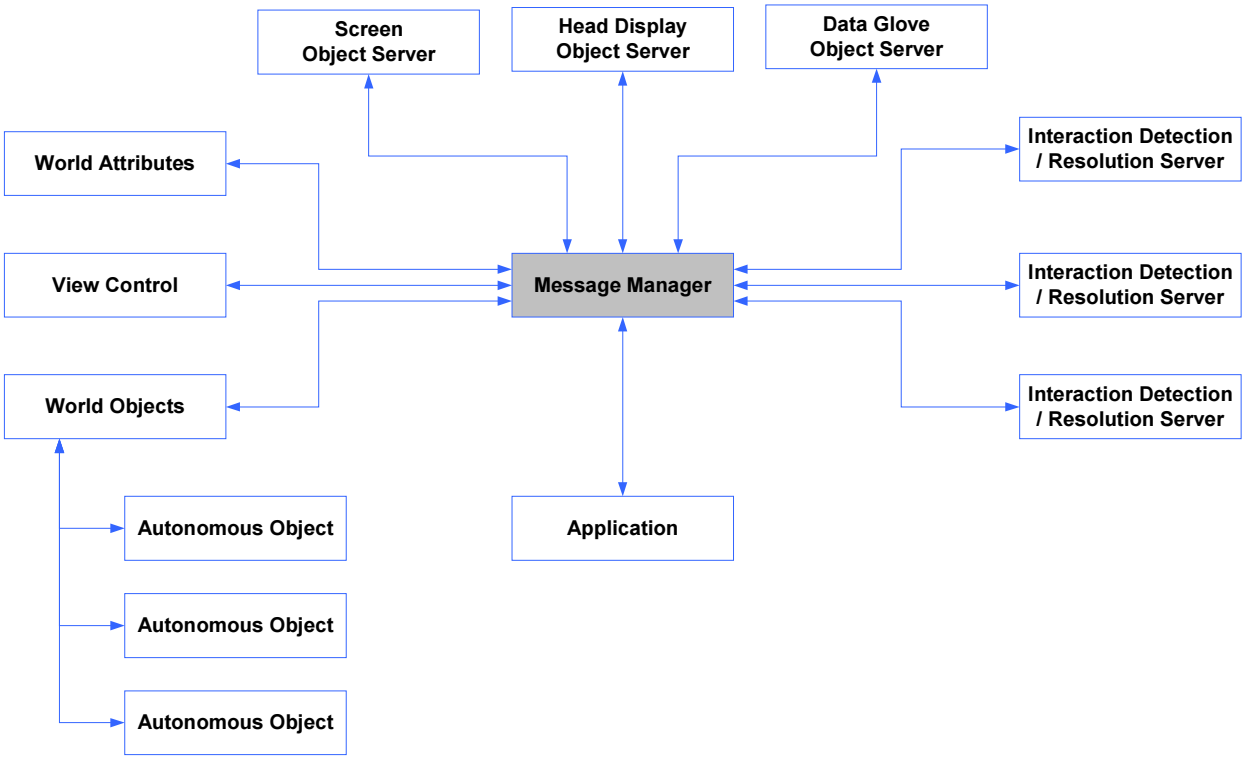


Abbildung 11: Softwarearchitektur eines Networked Virtual Environments

Interaktions-Paradigma von HIDRA:

- **Highly Independent Entities:** Die einzelnen Objekte der Umgebung sind vollständig unabhängig voneinander. Sie werden zentral verwaltet.
- **Locally Dependend Entities:** Die einzelnen Objekte der Umgebung hängen vom lokalen Kontext ab. Die Interaktion wird zentral verwaltet, jedoch lokal im Kontext aufbereitet.
- **Highly Dependend Entities:** Die einzelnen Objekte der Umgebung sind miteinander verknüpft. Diese Objekte werden lokal verwaltet und aufbereitet.

4 Multiplayer Computer Games

4.1 Einleitung

Multiplayer Computer Games (MCG) werden in den nächsten Jahren eine grosse Rolle in der Unterhaltungsindustrie spielen. Bereits heute werden die meisten Spielkonsolen mit denjenigen Netzwerkinterfaces ausgestattet, die für Desktopgeräte und Mobile Devices bereits etabliert sind.

Die zentralen Aspekte aus Architektursicht für solche Systeme sind:

- **Networking Issues:** Die Art und Weise, wie mit der Tatsache umgegangen wird, dass das Netzwerk die zentrale Limitation solcher System darstellt.
- **Scheduling:** Die Realisierung von Real-Time Interaktivität auf lowcost Endgeräte.
- **Referenzarchitekturen:** Der Versuch für bestimmte Game-Typen "Best Practice" Architekturen bereitzustellen.

4.2 Networking Issues

Die drei bestimmenden Faktoren für das Verhalten eines Netzwerkes sind:

- **Bandbreite**
- **Latenz**
- **Rechenleistung**

Die klassische Umgangsweise mit diesen Faktoren besteht darin, bestimmte Verteilungskonzepte zu realisieren. So wird anhand eines "Communication Graphs" der Freiheitsgrad der Verteilung definiert.

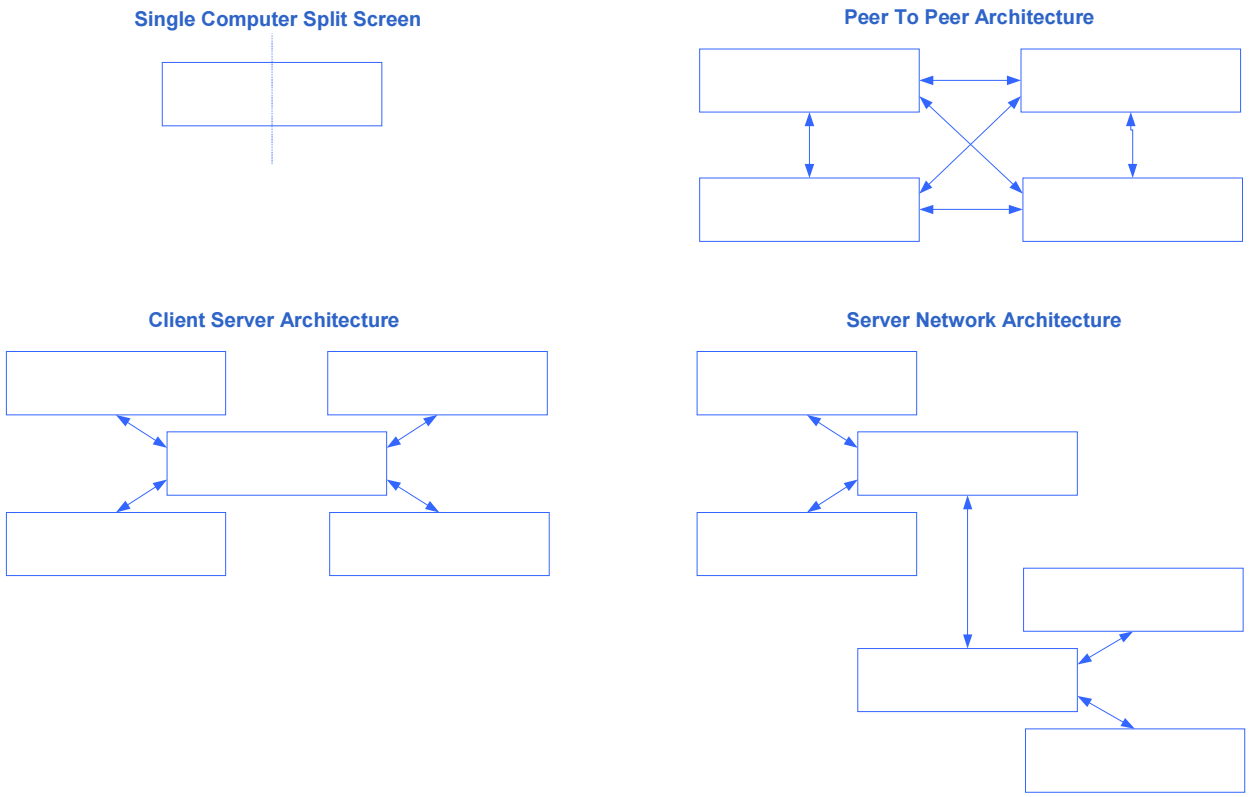


Abbildung 12: Verteilungskonzepte

Simulationen, Networked Virtual Environments und Multiplayer Computer Games versuchen jedoch die Limitationen eines Netzwerkes mit verschiedenen Techniken zu umgehen:

- **Message Compression and Aggregation:** Die Zusammenfassung, Komprimierung und oft auch Verschlüsselung von Meldungen verringert das Datenvolumen der Kommunikation.
- **Interest Management:** Diese Technik erlaubt es, lediglich diejenigen Informationen einer virtuellen Welt abzugleichen, die im betroffenen Kontext eines Clients gerade von Interesse sind (Area of Interest).
- **Dead Reckoning:** Basierend auf der Navigationstechnik und einem bekannten Startpunkt und einer bekannten Geschwindigkeit wird die Darstellung interpoliert, ohne dass ein Update über das Netz stattfinden muss.

4.3 Scheduling

Die Art und Weise des Scheduling bestimmt die Interaktivität eines Multiplayer Games, da in jedem solchen System eine gemeinsame Welt geteilt wird. Diese Teilung setzt den Einsatz einer zentralen Stelle (ein oder mehrere Server) voraus, die die Darstellung dieser Welt verwalten und die entsprechenden Informationen an alle beteiligten Clients senden. Die Art und Weise, wann und wie oft die Clients Updateinformationen erhalten, respektive Updates senden können, wird durch die Realisierung der entsprechenden Scheduling Algorithmen realisiert:

- **Lineares Scheduling** (Round Robin – First Come First Served) arbeitet in einem Distributed Real-Time Environment mit gleichmässigen Zeitintervallen für alle beteiligten Entitäten unabhängig vom Kontext des betroffenen Clients.
- **Priority Round-Robin Scheduling** arbeitet mit einer Multilevel Feedback Queue (MLFQ), um dynamisch die Abfolge ändern zu können.
- **Context Dependend Scheduling** beziehen sowohl den Zustand der darzustellenden Welt, wie auch die Situation der einzelnen Clients in den Scheduling Algorithmus ein. Typischerweise arbeiten diese Algorithmen mit sogenannten "Temporal Bounding Volumes" (TBV) und "Update Free Regions" (UFR) Technologien.

Ein **Temporal Bounding Volume** ist ein Bereich eines Raumes, der ein bestimmtes Objekt vollständig über eine bestimmte Zeitperiode hinweg enthält. Ein Objekt in einem nichtsichtbaren TBV muss nicht nachgeführt werden.

Die **Update Free Regions** Technik unterscheidet zwischen sich gegenseitig überlappenden Objekten. Ein Update ist nur in denjenigen Teilen der Objekte notwendig, die auch sichtbar sind.

4.4 Architekturen

4.4.1 Skalierbare Umgebung

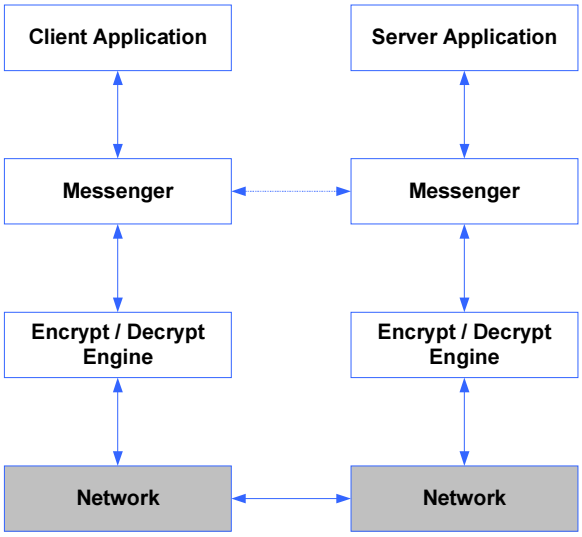


Abbildung 13: Layered Basis architecture

Server

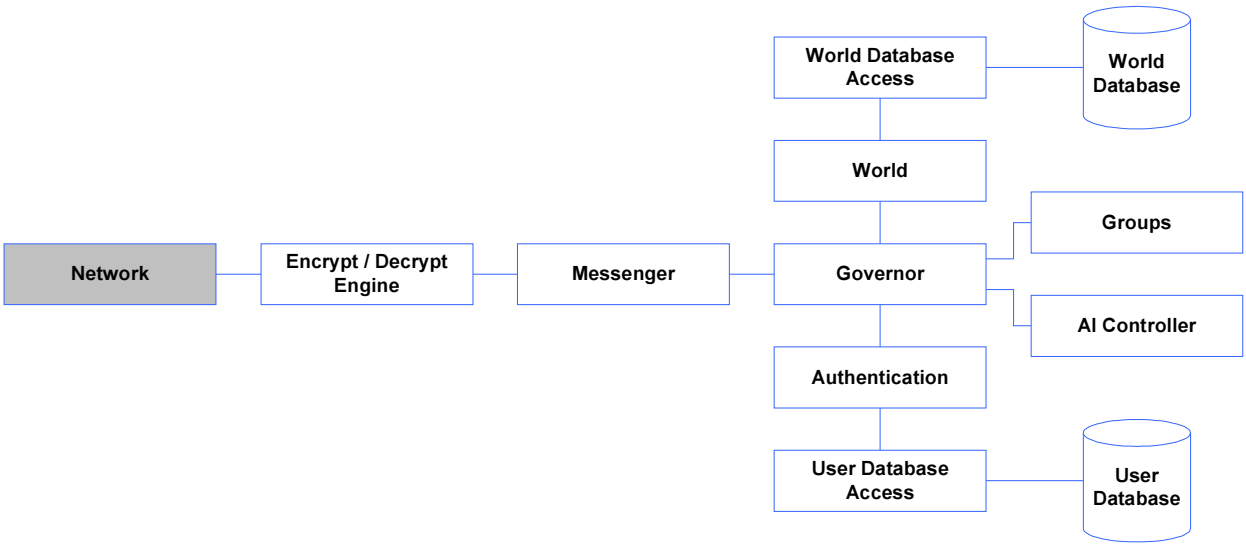


Abbildung 14: Beispiel eines Servers

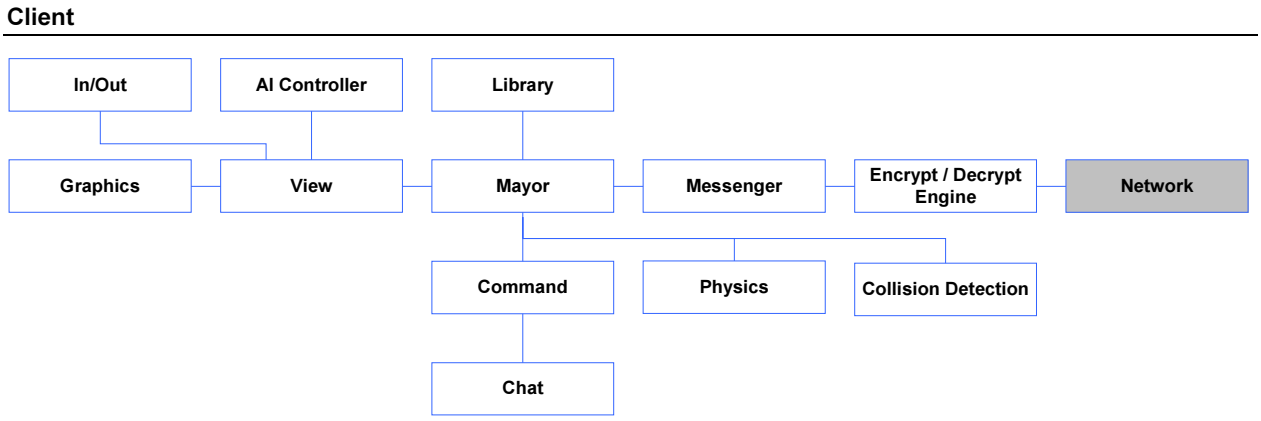


Abbildung 15: Beispiel eines Clients

Genereller Aufbau [Caltagirone et al. 2002]**:**

- **Communication Components:** Das Network, die Encrypt / Decrypt Engine sowie der Messenger sind für die Kommunikation zwischen dem Client und dem Server zuständig.
- **World Components:** World, World Database Access sowie World Database enthalten alle Informationen über die virtuelle Welt. Die darzustellende Welt selbst ist lediglich ein Subset der World Database, die die gesamte virtuelle Welt enthält.
- **User Components:** Die Authentication, User Database Access sowie die User Database enthalten alle Informationen über den Status und die Rechte eines bestimmten Users.
- **Groups Component:** Das Groups Module enthält alle Informationen (Regeln, Mitglieder, etc.) über eine Gruppe, der ein bestimmter User angehört.
- **AI Component:** Die Logik für die virtuellen Spieler, die zu verschiedenen Zeitpunkten in das System eingebracht werden.
- **Governor:** Komponente zur globalen Steuerung des Spiels.
- **Maior:** Lokale Komponente zur Steuerung des Spiels.

4.4.2 Augmented Reality Umgebung

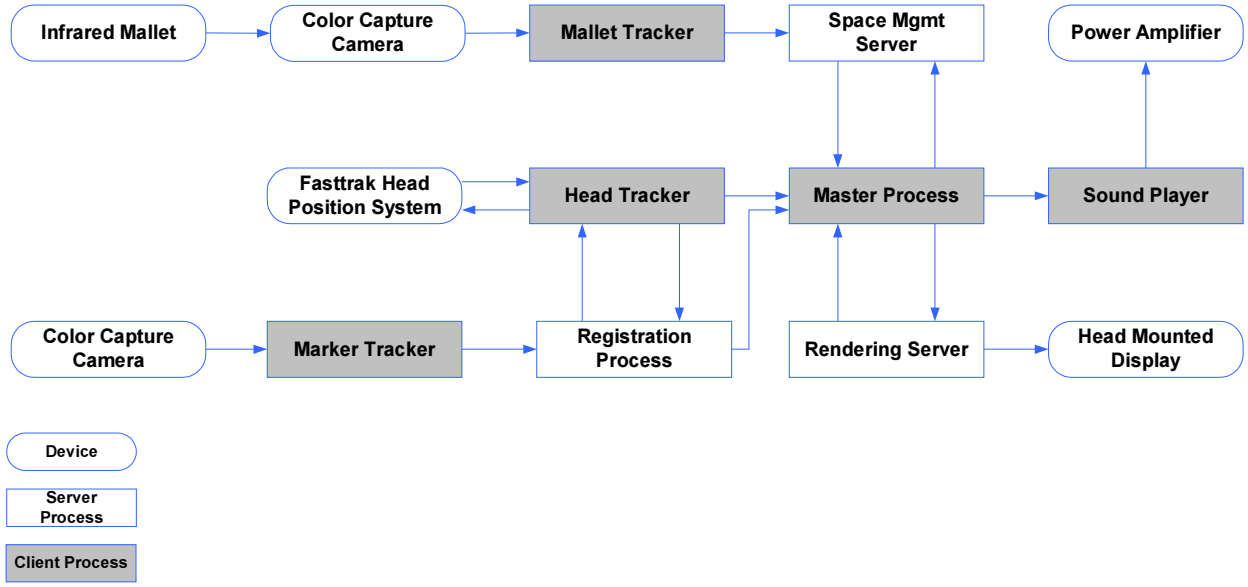


Abbildung 16: Beispiel eines Augmented Reality Systems (Hockey Game [Ohshima et al. 1998])

Ein Bereich der Distributed Real-Time Systems und deren Game-Ausprägungen befasst sich mit der detailgetreuen Nachbildung der Realität. Der Spieler / die Spielerin soll sich als beteiligte Person fühlen, die dasselbe erlebt, wie ein Hochleistungssportler es erlebt. Diese Systeme arbeiten normalerweise mit Motion-Tracking und Headmounted-Displays oder Data-Suits.

5 Exkurs: Hardware-Architektur der Game-Konsolen

5.1 Einleitung

Die am meisten verwendeten Spiel-Konsolen sind (Verkaufszahlen gemäss vgchartz.com bis Ende November 2015):

Name	Hersteller	Anzahl Einheiten
PlayStation 2 (PS2)	Sony	157 Millionen
Nintendo DS (Dual Screen)	Nintendo	154 Millionen
Game Boy (GB)	Nintendo	118 Millionen
PlayStation (PS)	Sony	104 Millionen
Wii (Wii)	Nintendo	101 Millionen
PlayStation 3 (PS3)	Sony	86 Millionen
Xbox 360 (X360)	Microsoft	85 Millionen

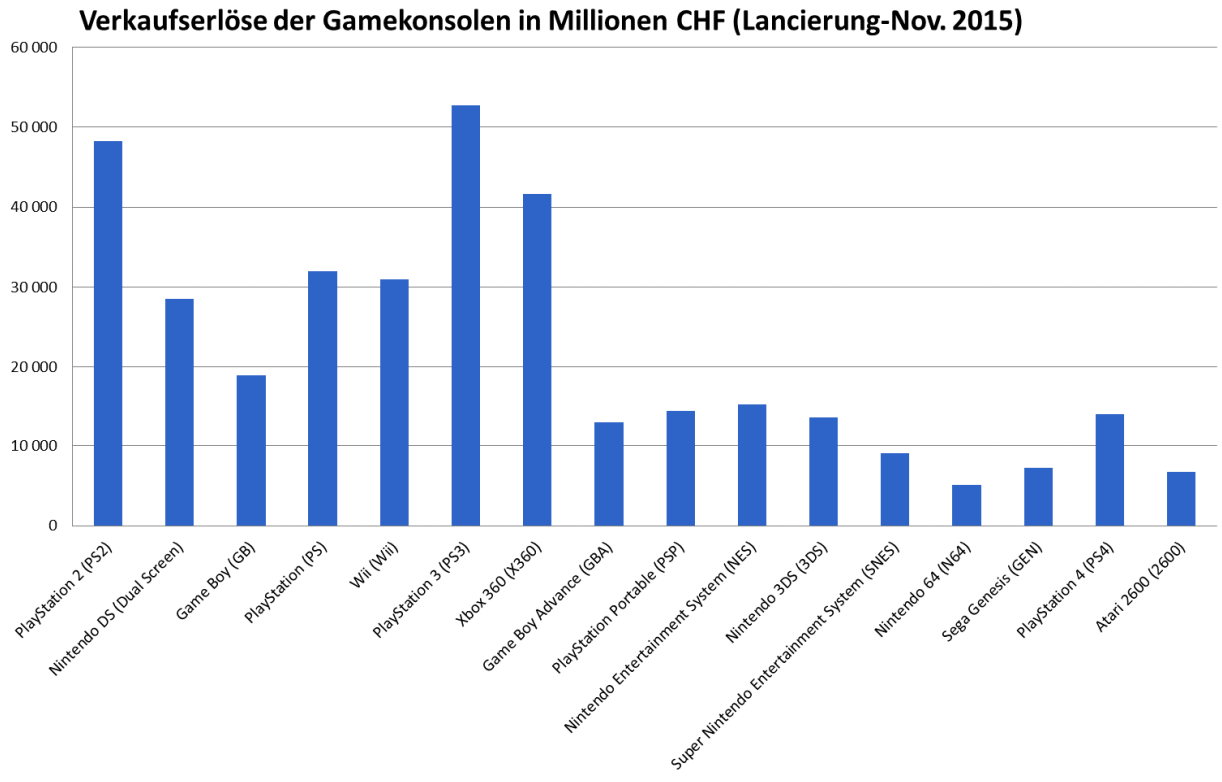


Abbildung 17: Verkaufserlöse mit Gamekonsolen in Millionen CHF

Mit Gamekonsolen wurde in den letzten 20 Jahren über 330 Mrd CHF verdient. Die grossen Hersteller Sony, Nintendo und Microsoft dominieren den Markt. Sony hat 160 Mrd CHF mit allen PlayStation Varianten verdient, gefolgt von Nintendo mit einer Vielzahl verschiedener Konsolen 113 Mrd CHF und Microsoft Xbox 52 Mrd CHF.

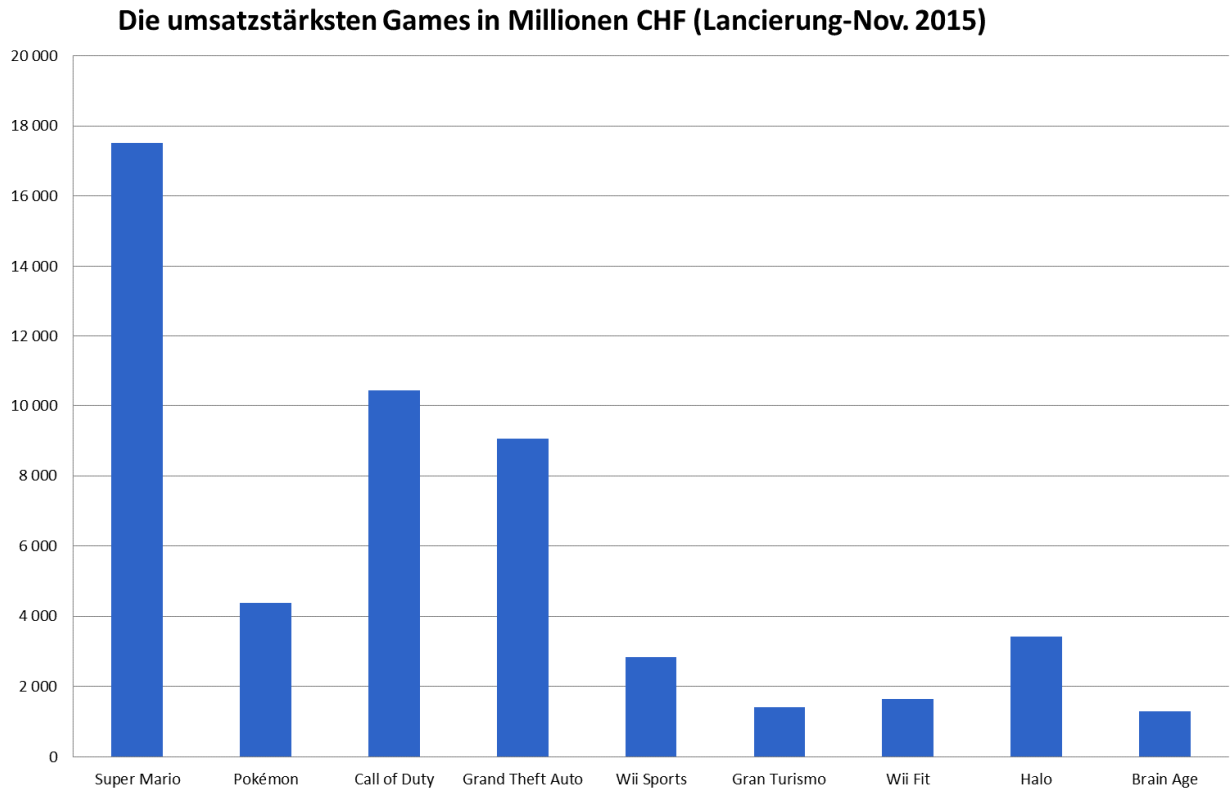


Abbildung 18: Verkaufserlöse mit Games in Millionen CHF

Die am meisten verkauften Spiele sind (Verkaufzahlen gemäss vgchartz.com bis Ende November 2015):

Platz	Name	Herausgeber	Anzahl Einheiten
1	Super Mario	Nintendo	357.28
2	Pokémon	Nintendo	178.59
3	Call of Duty	Activision	142.11
4	Grand Theft Auto	Take-Two Interactive	123.34
5	Wii Sports	Nintendo	115.33
6	Gran Turismo	Sony Computer Entertainment	57.73
7	Wii Fit	Nintendo	44.52
8	Halo	Microsoft Game Studios	39.87
9	Brain Age	Nintendo	35.45
10	Final Fantasy	Sony Computer Entertainment	33.84
11	Tetris	Nintendo	30.26
12	Duck Hunt	Nintendo	28.31

5.1.1 Vergleich Games & Filmindustrie

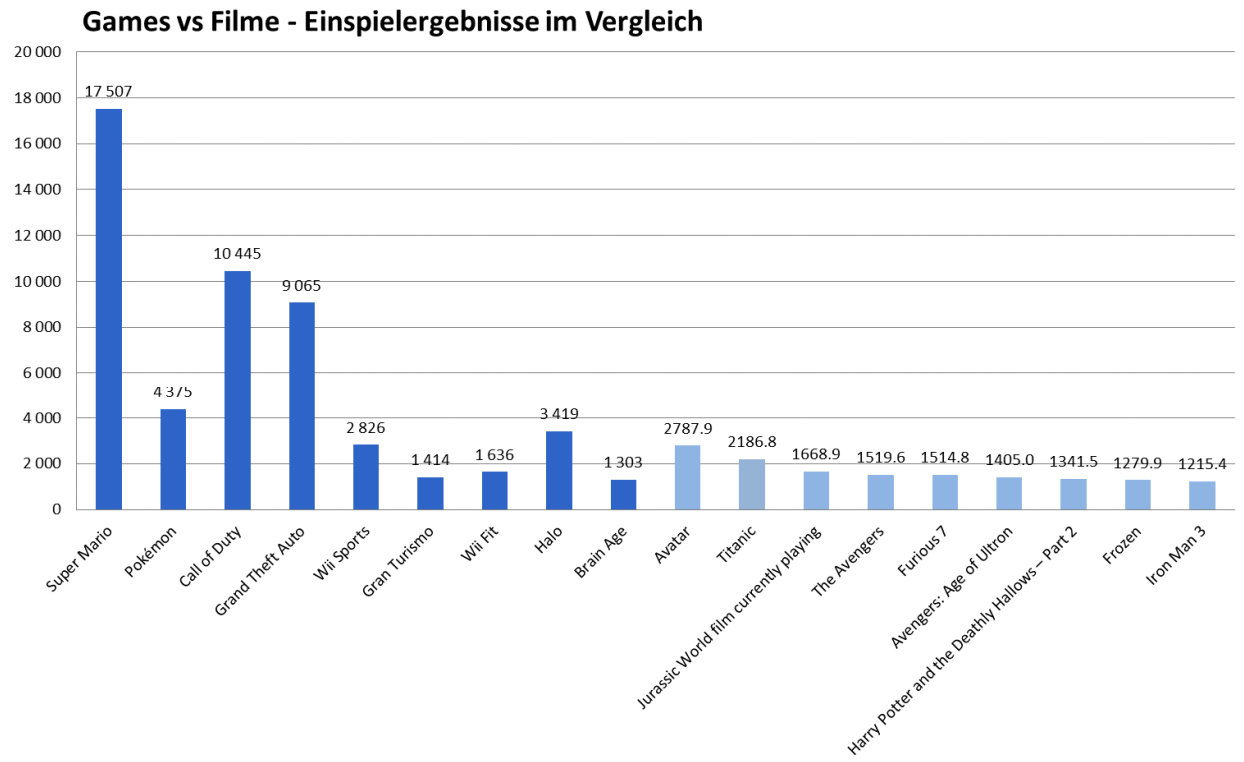


Abbildung 19: Verkaufserlöse mit Gamekonsolen in Millionen CHF

Die 9 umsatzstärksten Computerspiele im Vergleich mit den 9 erfolgreichsten Kinofilmen.

5.2 Playstation 2

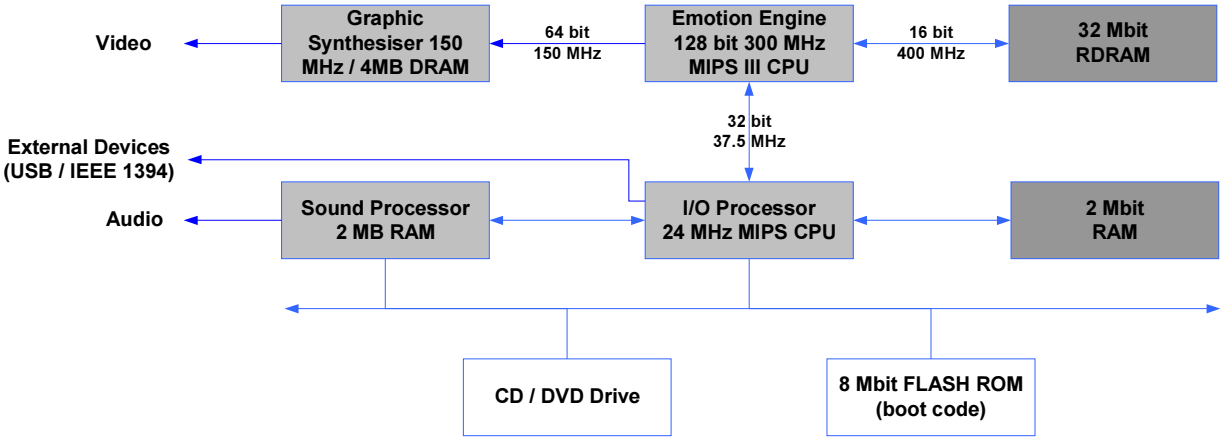


Abbildung 20: PS2 Hardware Architektur

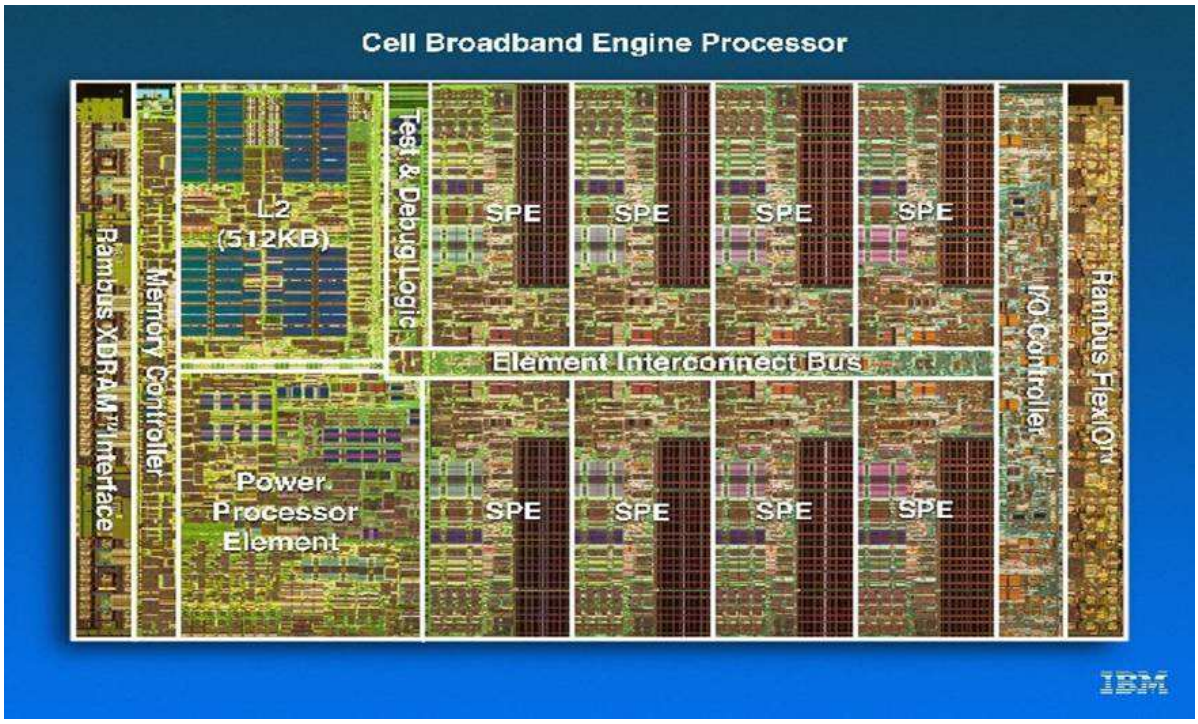
Die PlayStation 2 ist mit über 157 Millionen verkauften Einheiten nach wie vor die meistverkaufte Game Console der Welt. Innovatives Kernstück der Console ist die Emotion Engine, ein spezialisierter 3D Prozessor für Spiele. Die

Emotion Engine produziert so genannte "Display Lists", Rendering Command Sequenzen, die an den Graphic Sythesiser gesendet werden. Die Emotion Engine führt sowohl geometrische Berechnungen als auch Simulations-Berechnungen durch.

Technische Daten:

- **CPU:** 128 Bit Emotion Engine, Taktfrequenz: 294.912 MHz Eingebauter VRAM Cache: 4 MB
- **SPU2:** Anzahl Stimmen: 48 plus Software Sound-Speicher: 2 MB
- **IOP:** I/O Processor CPU inkl. Play Station CPU Taktfrequenz: 33.8688 MHz oder 36.864 MHz (auswählbar) IOP Speicher: 2 MB
- **Laufwerk:** CD-ROM und DVD-ROM, Geschwindigkeit: CD-ROM 24-fach, DVD-ROM 4-fach
- **Anschlüsse:** 2 Controller Ports, 2 Memory Card Steckplätze, AV Multikabel Ausgang (Optical digital Ausgang), 2 USB Ports, IEEE1394 i-Link

5.3 PlayStation 3



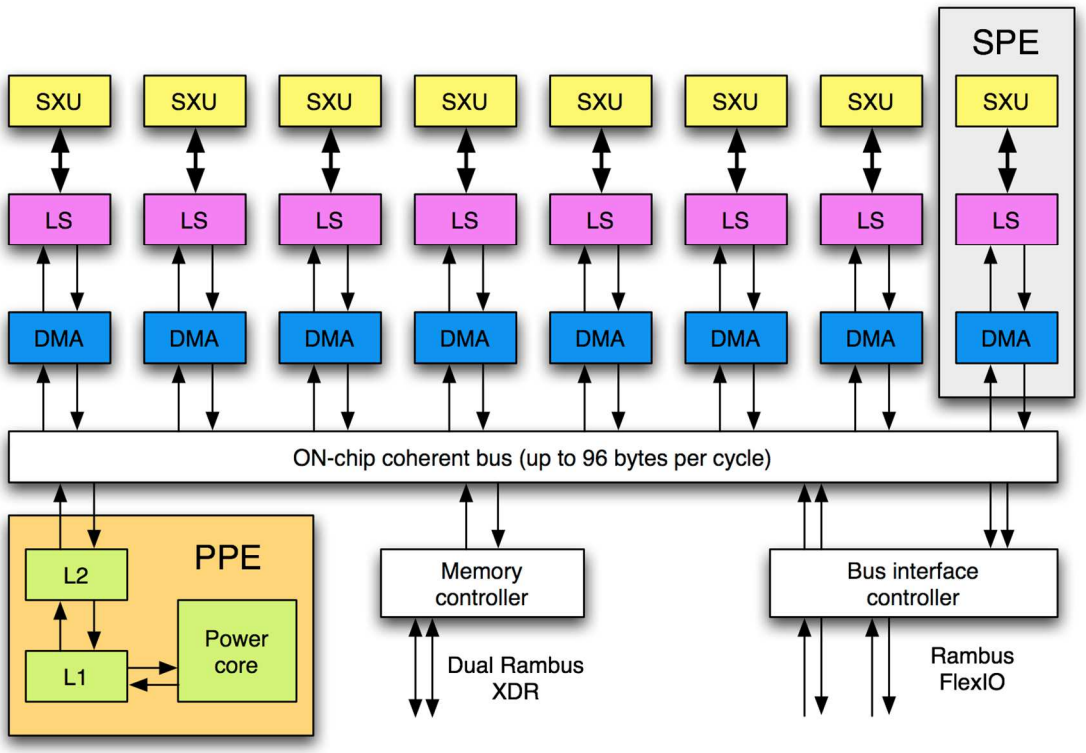


Abbildung 21: PlayStation 3 Cell Processor

- **SPE:** Synergistic Processor Element
- **LS:** Local Storage (256 Kb pro SPE)

- **EIB:** Element Interconnection Bus
- **PPE:** Das Power Processing Element in ein 64-bit PowerPC-5 RISC-Prozessor der die zentrale Steuerung übernimmt

Die Playstation 3 basiert auf dem IBM Cell Processor. Die Prozessortechnologie hat eine Leistung von 1 Teraflop. Und wird in Supercomputern wie dem Roadrunner von IBM verwendet.

5.4 XBOX

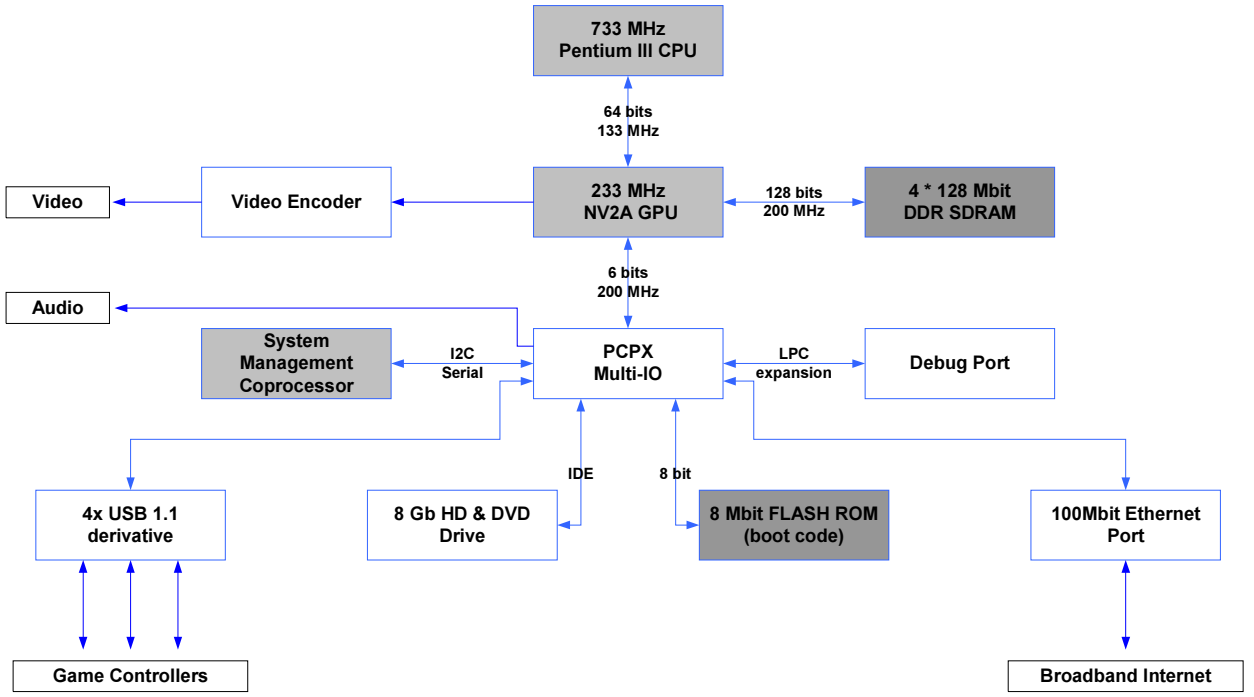


Abbildung 22: Xbox Hardware Architektur

Die Xbox wurde Ende 2001 in den USA auf den Markt gebracht und bis Ende 2006 wurden ca. 24.5 Millionen Einheiten weltweit verkauft. Zur Unterstützung des Markteintrittes von Microsoft ist gleichzeitig mit der Xbox das Spiel Halo: Combat Evolved lanciert worden, welches speziell auf die Spielkonsole zugeschnitten worden war (Microsoft hatte den Hersteller des Spieles gekauft).

Das Spiel selbst hat in seiner 2. Version – Halo 2 Geschichte geschrieben, da bereits am ersten Tag 3.5 Millionen Einheiten Verkauft worden sind. Damit hat zum ersten Mal in der Geschichte ein Computerspiel mehr eingebracht als je ein Kinofilm zuvor. Ein Meilenstein für die Informatik.

Technische Daten:

- Gesamtleistung 12 Gigaflops
- Intel CPU mit 733 MHz und 64 MB RAM.
- Geoforce 3 Graphic von Nvidia mit 233 MHz und einer Auflösung von 1920x1080 Pixel
- Ethernet 10/100 Mbit
- 8 oder 10 GB Harddisk

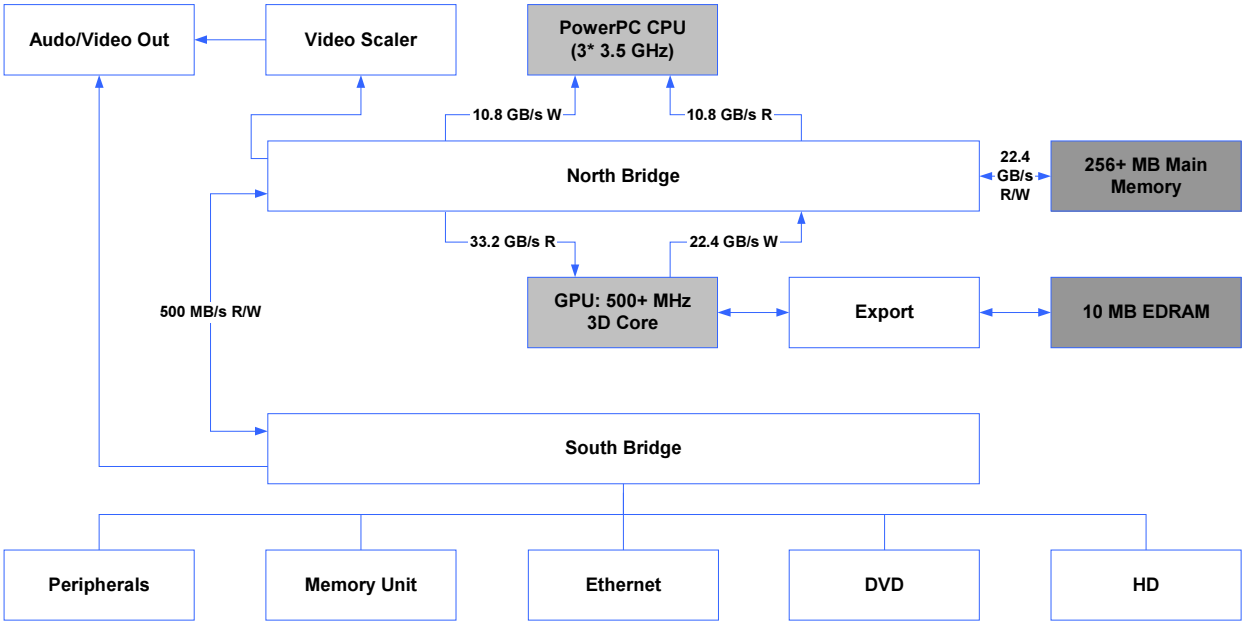


Abbildung 23: Xbox 360 Hardware Architektur

Die XBOX 360 Console ist die zweite Generation der Engine von Microsoft, die Ende 2005 auf den Markt gekommen ist. Es sind bis heute 83 Millionen Einheiten verkauft worden. Bei Einführung war die Konsole zwar ein sehr grosser Erfolg, Microsoft schrieb aber bis ins Jahr 2008 Verluste in Milliardenhöhe, da die Xbox 360 nicht kostendeckend produziert werden konnte. Dafür war sie jedoch sehr leistungsfähig.

Technische Daten:

- Gesamtleistung 77 Gigaflops
- Die Konsole basiert auf einer „triple-core“ IBM PowerPC Prozessor. Jeder der drei Prozessoreinheiten ist mit 3.5 GHz getaktet und hat je einen 32KB Level1 Instruction Cache und einen 32KB Level1 Data Cache. Ausserdem teilen sich die drei Einheiten einen 1MB Level2 Cache.
- Die GPU (Graphic Processor Unit) ist ein ATI Prozessor mit einer „peak pixel rate“ von 4 Gigapixels/s und einer OnChip Realiserierung von Direct3D 9.0.
- 512 MB RAM und Festplatten bis 320 GB

5.5 Die neuste Generation

Abbildung 24: GPU Hardware Architektur von Xbox und PlayStation

Die neuste Generation der Spielkonsolen Xbox (PowerPC) und PlayStation (Cell) gleichen sich in Sachen Hardware-Architektur wieder an, was für die Game-Entwicklung eine grosse Vereinfachung darstellt. Der Wechsel der zur x86-Technik und einer identischen GPU (Grafical Processing Unit verkleinert den Aufwand für Programmierung, Optimierung und Support von Spielen.

6 Anhang A: High-Level Development of Multiserver Online Games