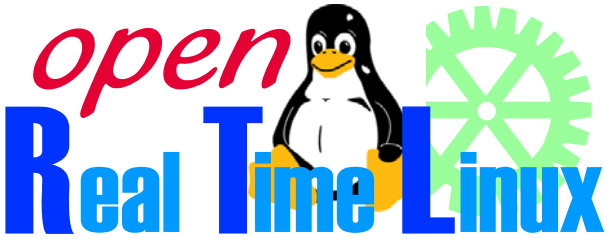
Fraunhofer Institut Experimentelles Software Engineering

RTLOpen Plattform Handbuch
Teil 5 – Entwicklungswerkzeuge

Autoren:
Armin Mück (Berghof)
Johann Wittner (Hofmann)
Ralf Kalmar (IESE)
Samir Amiry (IESE)
Thorsten Keuler (IESE)
Mario Kühnemund (VisionTools)



IESE-Report Nr. 057.07/D
RTLOpen-Report 006/D
Version 1.0
1. September 2007

Eine Publikation des Fraunhofer IESE

Eine Veröffentlichung des RTLOpen-Projekts FKZ 01 IS C14

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen FKZ 01 IS C14 gefördert.

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.
Das Institut transferiert innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach (geschäftsführend)
Prof. Dr. Peter Liggesmeyer
Fraunhofer-Platz 1
67663 Kaiserslautern

Abstract

In diesem Bericht werden in der Praxis bewährte, frei verfügbare Software-Werkzeuge zur Entwicklung von (eingebetteten) Realzeit-Systemen charakterisiert und vorgestellt. Dabei werden für jede Entwicklungsaktivität ein oder zwei Tools vorgestellt, so dass eine vollständige Toolkette aufgebaut werden kann.

Schlagworte: Linux, RTAI, tool chain, real-time, RTLOpen

Copyright © Fraunhofer IESE 2007

Inhaltsverzeichnis

1	**Einleitung**	**1**
1.1	Hintergrund und Motivation	1
1.2	Wer sollte diesen Bericht lesen?	2
1.3	Konzeption und Struktur des Berichts	2
1.4	RTLOpen Toolkette	3
2	**Infrastruktur**	**4**
2.1	Konfigurationsmanagement	4
2.1.1	CVS	4
2.1.2	Subversion	5
3	**Anforderungsspezifikation**	**7**
4	**System-Entwurf**	**9**
4.1	System-Modellierung	9
4.1.1	Scilab	9
4.1.2	Poseidon (UML)	10
5	**Realisierung**	**12**
5.1	Integrierte Entwicklungsumgebungen (IDE)	12
5.1.1	KDevelop	12
5.1.2	Eclipse	13
5.1.3	Source Navigator	14
5.1.4	Kylix	15
5.1.5	RTAI Lab	15
5.2	Compiler und Cross-compiler	17
5.2.1	GNU Compiler Collection (GCC)	17
5.3	Source Code Dokumentation	17
5.3.1	Doxygen	17
5.4	Debugger	19
5.4.1	GNU Debugger	19
5.4.2	Data Display Debugger (ddd)	20
5.4.3	Lauterbach	21
5.4.4	BDI 2000	22
6	**Systemintegration**	**23**
6.1	Build-Tools	23
6.1.1	Ant und Anthill	23
6.1.2	GNU Make	24

Copyright © Fraunhofer IESE 2007

7 Qualitätssicherung 25
7.1 Testing Tools 25
7.1.1 Gcov Coverage Tool 25
7.1.2 CppUnit 26
7.2 Analyse-Werkzeuge 26
7.2.1 Linux Trace Toolkit 26

8 Beispiele für Toolketten 27
8.1 Entwicklung eingebetteter Linux Realtime-Systeme (Berghof) 27
8.2 Entwicklung eines RTAI Subsystems (Hofmann) 30
8.3 Linux Anwendungsentwicklung mit Realzeit-Komponenten (VisionTools) 32

Copyright © Fraunhofer IESE 2007

1 Einleitung

Dieses Dokument ist Teil 5 des RTLOpen Plattform Handbuchs.

Die RTLOpen Plattform Dokumentation:

Teil 1: Übersicht

Teil 2: Plattform Architektur

Teil 3: Softwareentwickung von Echtzeitsystemen

Teil 4: Echtzeit-Linux

Teil 5: Entwicklungswerkzeuge (dieses Dokument)

Teil 6: Plattform Demonstrator und Beispiel

Zusätzliche Information findet man auf der Projekt-Homepage: www.open-realtime-linux.de

Das Projekt RTLOpen (Open Realtime Linux für den Maschinenbau) entwickelt eine Referenzplattform die flexibel, hoch performant und zukunftssicher ist. Sie basiert auf Open-Source Komponenten wie beispielsweise dem Betriebssystem Linux. Die RTLOpen Plattform ist in einem Handbuch dokumentiert das aus verschiedenen Teilen besteht. (s. oben)

Das Projekt ist Teil der Deutschen Forschungsinitiative "Software Engineering 2006" (SE-2006) des Bundesministeriums für Bildung und Forschung. Die Projektnummer ist 01 IS C 14.

1.1 Hintergrund und Motivation

Software-Werkzeuge unterstützen Entwickler in vielfältiger Hinsicht. Jedoch ist die Auswahl des richtigen Werkzeugs für die unterschiedlichen Aktivitäten nicht einfach und kann zu Problemen bei Einhaltung von Auslieferungszeit, Projektkosten und Produktqualität führen.

OpenSource-Software hat den Reiz, in der Anschaffung sehr kostengünstig zu sein und in Bezug auf die individuelle Anpassbarkeit alle Möglichkeiten zu liefern. Allerdings fehlt die Haftung des Herstellers und steigert somit das eigene Risiko. Zusätzlich muss mit einem erhöhten Einarbeitungsaufwand gerechnet werden, da vielfach keine Schulungen oder Sekundärliteratur verfügbar sind.

Copyright © Fraunhofer IESE 2007

Bei der Vielfalt der OpenSource-Projekte ist es schwierig, den Überblick zu bewahren. Insbesondere für den speziellen Fall der Realzeit-Software-Entwicklung kommt hinzu, dass das Angebot sehr begrenzt ist.

In diesem Bericht werden in der Praxis bewährte, frei verfügbare Software-Werkzeuge zur Entwicklung von (eingebetteten) Realzeit-Systemen charakterisiert und vorgestellt. Dabei werden für jede Entwicklungsaktivität ein oder zwei Tools vorgestellt, so dass eine vollständige Toolkette aufgebaut werden kann.

1.2 Wer sollte diesen Bericht lesen?

Dieser Bericht richtet sich an alle Personen, die Software-Werkzeuge zur Entwicklung von (eingebetteten) Realzeit-Systemen auf Basis der RTLOpen-Plattform einsetzen möchten. Durch die offene Spezifikation der RTLOpen-Plattform können die hier genannten Werkzeuge für viele Linux und RTAI-Linux-Systeme verwendet werden.

1.3 Konzeption und Struktur des Berichts

Dieser Bericht bietet eine nach Entwicklungstätigkeiten sortierte Sammlung von bewährten und frei verfügbaren Software-Werkzeugen. Jedes Werkzeug wird dabei in einer tabellarischen Form charakterisiert und beschrieben. Sämtliche hier vorgestellte Werkzeuge wurden im Rahmen des Projekts RTLOpen evaluiert und bilden zusammen einen Baukasten, aus dem eine vollständige Toolkette zur Entwicklung von Software für Realtime Linux zusammengestellt werden kann. Beispiele für solche sich im realen Einsatz befindliche Toolketten werden in Kapitel 8 gegeben.

Bei der Auswahl wurde besonderer Wert auf Praktikabilität und Stabilität der Werkzeuge für den Einsatz in realen Industrieprojekten gelegt.

Beispiel-Tabelle für Werkzeug-Charakterisierung

Name und Version	gcc 3.4.4 (GNU Compiler Collection)
Lizenz / Preis	OpenSource, GPL, frei im Internet verfügbar
Zweck	Standard-Linux-Compiler für C/C++ und Prozessorarchitekturen.
Vorteile	• Inline Assembler • C und C++ Unterstützung
Erfahrungen	Ältere Versionen sehr ausgereift.

Copyright © Fraunhofer IESE 2007

	Es lohnt sich nicht, immer die neuste Version zu nutzen, da diese alten Code u.U. anders übersetzt.
Konfigurationstipps	Es lohnt sich meist kaum, gcc selbst zu übersetzen und zu installieren, wenn man keine bestimmte Version benötigt. Gcc ist in allen Linux-Distributionen enthalten.
Bezugsquelle	http://gcc.gnu.org/

1.4 RTLOpen Toolkette

Die RTLOpen-Toolkette versteht sich als modularer Baukasten, der es ermöglicht für verschiedene Kontexte eine geeignete Toolkette aufzubauen. Die Werkzeuge sind in unterschiedliche Kategorien gemäß des Software-Entwicklungsprozesses gruppiert, wie nachfolgende Abbildung zeigt:

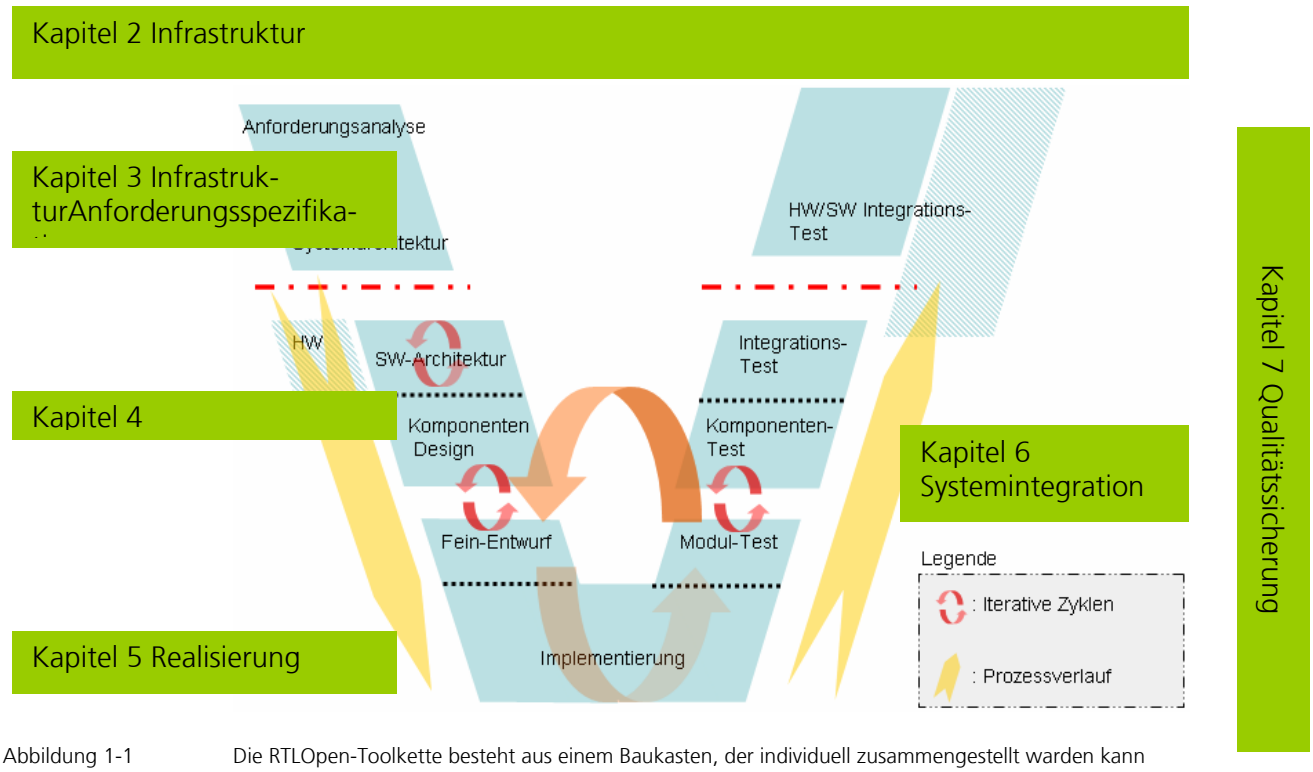


Abbildung 1-1 Die RTLOpen-Toolkette besteht aus einem Baukasten, der individuell zusammengestellt warden kann

Beispiel-Implementierungen sind in Kapitel 8 angegeben.

Copyright © Fraunhofer IESE 2007

2 Infrastruktur

In diesem Kapitel sind allgemeine unterstützende Werkzeuge für die Entwicklungsinfrastruktur aufgeführt.

2.1 Konfigurationsmanagement

Konfigurationsmanagement spielt eine wichtige Rolle in der Entwicklung von Sofwaresystemen. Es sind dabei nicht nur Umstände wie verteilte Entwicklung oder parallele Entwicklung mit vielen Teams die den Einsatz eines Konfigurationswerkzeugs empfehlen. Bei der Entwicklung von Softwaresystemen sollte man zu jeder Zeit in der Lage sein den Entwicklungsstand eines gewissen Zeitpunktes rekonstruieren zu können. Dabei ist es wichtig nicht nur die relevanten Dokumente zu finden, sondern ebenso alle nicht relevanten zu filtern.

2.1.1 CVS

Name und Version	Concurrent Versioning System (CVS) Version 1.11.22
Lizenz / Preis	GNU General Public License, V2, OpenSource
Zweck	Versionsbasierte Dokumentenverwaltung
Vorteile	• weit verbreitet, bewährt und stabil • nutzt RCS-Standard, mit komfortablen Erweiterungen:
Erfahrungen	
Konfigurationstipps	
Bezugsquelle	http://www.nongnu.org/cvs/
Kommerzielle Pendants	MS SourceSafe

Abbildung 2-1 zeigt die Einträge im Kontextmenüs von CVS angewandt auf das Dateisystem des Betriebssystems.

Copyright © Fraunhofer IESE 2007

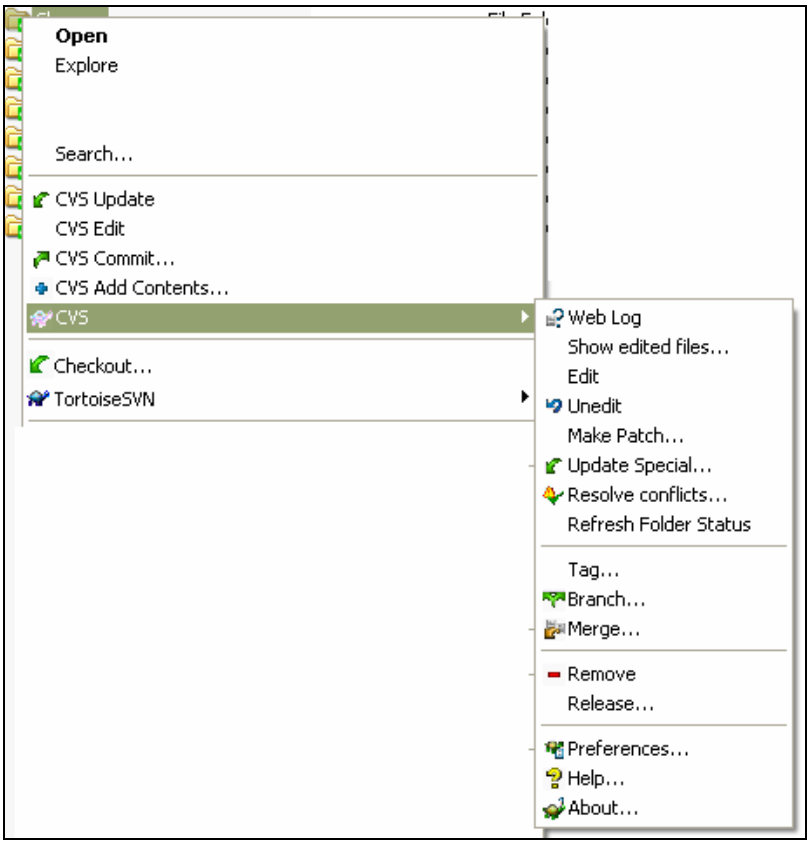


Abbildung 2-1 Screenshot des CVS Kontextmenüs

2.1.2 Subversion

Name und Version	Subversion V1.4
Lizenz / Preis	Apache/BSD-Lizenz, OpenSource
Zweck	Versionsmanagement/Konfigurationsmangement von Dateien und Ordnern im Filesystem. Designierter Nachfolger des älteren „cvs"-Systems.
Vorteile	• versioniert auch Verzeichnisse und Umbenennungen • einfacher Zugang von jedem Entwickler-PC über das Netzwerk möglich (Apache-Server, standalone server-Option) • effizient auch für Binärdateien (binary diff) • Eclipse-Plugin „Subclipse" für die Integration in Eclipse verfügbar: http://subclipse.tigris.org/

Copyright © Fraunhofer IESE 2007

Erfahrungen	umfangreiche Dokumentation unter http://svnbook.red-bean.com/ , online Version (HTML und pdf) eines kommerziellen Buchs
Konfigurationstipps	
Bezugsquelle	http://subversion.tigris.org/

Copyright © Fraunhofer IESE 2007

3 Anforderungsspezifikation

In diesem Kapitel sind Werkzeuge aufgeführt, die die Phase der Anforderungsspezifikation unterstützten.

Für die Spezifikation der Anforderungen in Form von Lasten- oder Pflichtenheften warden in der Regel Textdokumente verwendet, so dass als Werkzeug z.B. Openoffice in Frage kommt [@Openoffice].

In manchen Branchen, wie z.B. in der Automobil- oder Luftfahrtindustrie haben sich mittlerweile spezielle Requirements-Management-Werkzeuge etabliert, die Anforderungen als einzelnes Datum verwalten und Beziehungen zwischen Anforderungen oder Änderungen an einzelnen Anforderungen verfolgbar gestalten. Auch hier gibt es mittlerweile Open Source Lösungen.

Name und Version	OSRMT V1.5 (Open Source Requirements Management Tool)
Lizenz / Preis	GPL, OpenSource
Zweck	Requirements Management.
Vorteile	• Implementierung in Java, dadurch Plattform-unabhängig • Mehrbenutzer-tauglich durch Web-Client • Versionskontrolle der Anforderungen • Binärobjekte möglich (z.B. für Zeichnungen) • Datenbank-basiert (nutzt verschiedene Datenbanken, wie z.B. mysql) • Verfolgung von Anforderungen mittels Traceability-Matrix • Anforderungsdokument wird automatisch generiert (Report-Generator) • Verwaltung von Änderungsanforderungen (Change Requests) • XML-Schnittstelle
Erfahrungen	Keine Erfahrungen im Maschinenbau vorhanden
Konfigurationstipps	
Bezugsquelle	http://www.osrmt.com/

Copyright © Fraunhofer IESE 2007

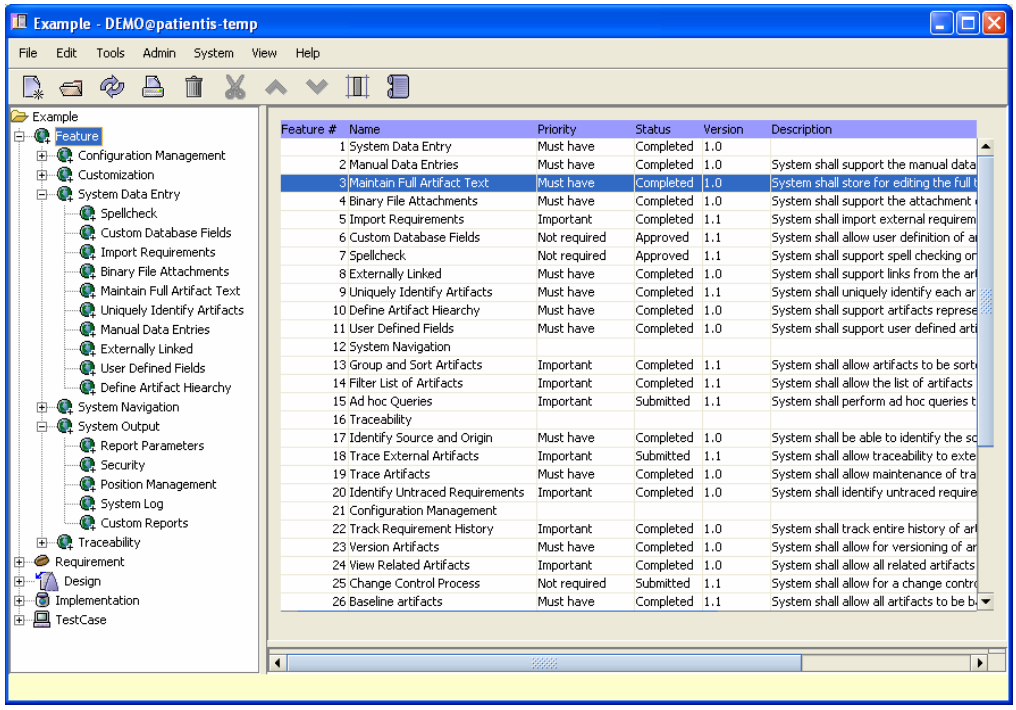


Abbildung 3-1 Verwaltung einzelner Anforderungen in OSRMT

Copyright © Fraunhofer IESE 2007

4 System-Entwurf

Dieses Kapitel enthält Werkzeuge zur Unterstützung des System-Entwurfs.

4.1 System-Modellierung

4.1.1 Scilab

Name und Version	Scilab 4.1 / Scicos
Lizenz / Preis	SCILAB Lizenz, Kostenlos
Zweck	Modellierungswerkzeug zur Entwicklung von Steuer- und Regelsystemen sowie Signalverarbeitungssysteme auf Basis von kontinuierlichen Modellen. Scicos ist ein Modellierungs- und Simulationswerkzeug (ähnlich Simulink). Scilab ein mathematisches Visualisierungswerkzeug (ähnlich Matlab).
Vorteile	• Graphische Modellierung, ähnlich wie Matlab/Simulink • Automatische Code Generierung mit RTAI Lab möglich
Erfahrungen	• Dank der mitgelieferten Demos aus allen Bereichen ist scilab schnell zu erlernen • Die Syntax ist Matlab-ähnlich – allerdings unterscheiden sich die Kommandos gelegentlich • Übertragungsfunktionen von Reglern können mit verschiedenen Plots (Nyquist, Wurzelortskurve, Bode, Impuls-, Sprungantwort) gut analysiert werden • Als Einführung eignet sich das Buch "Modeling and Simulation in Scilab/Scicos" von Stephen L. Campbell, Jean-Philippe Chancelier, and Ramine Nikoukhah (Springer Verlag).
Konfigurationstipps	• Komplexe Probleme formuliert man am besten mit einem Editor seiner Wahl (der scilab Editor ist gewöhnungsbedürftig) in einer externen Datei, die man dann in scilab mit dem Kommando exec(„dateiname") ausführen kann. • Eine gute Einführung und Vergleich mit Matlab im Bereich Systemanalyse findet man unter http://web.mit.edu/6.302/www/compare/report.html

Copyright © Fraunhofer IESE 2007

Bezugsquelle	http://www.scilab.org/ http://www.scicos.org/

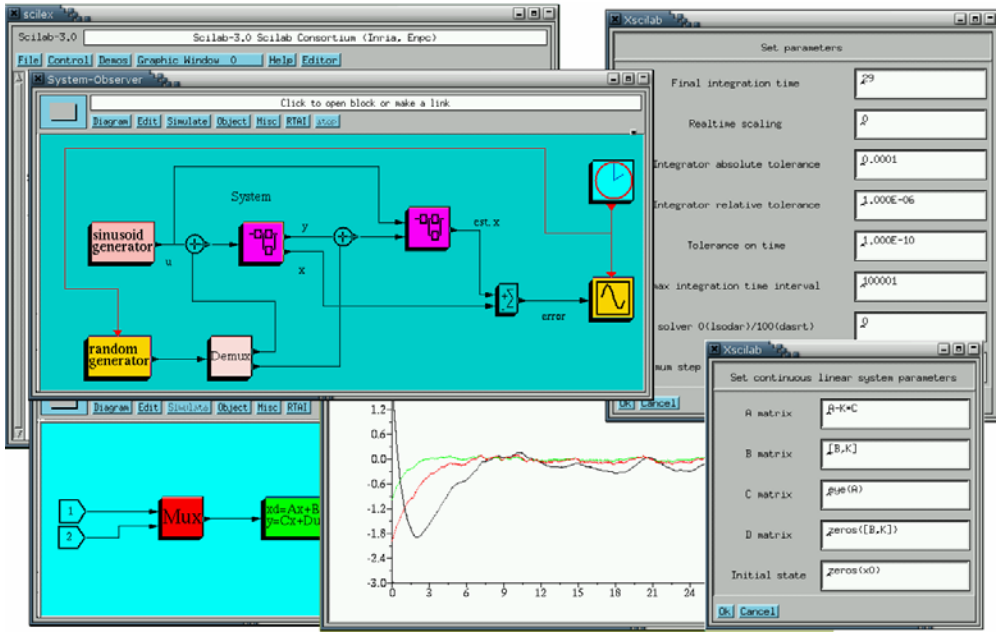


Abbildung 4-1 Grafische Systemmodellierung mit Scicos

4.1.2 Poseidon (UML)

Name und Version	Poseidon Community Edition v4.2
Lizenz / Preis	kostenlos
Zweck	UML Modellierungswerkzeug
Vorteile	• Implementiert in Java, Plattform-unabhängig • Speicherung im kompatiblen Format zum UML 2.0 Diagram Interchange Standard • Unterstützt XMI 1.2 als Standard Speicherungsformat. XMI 1.0, 1.1 und 1.2 können geladen werden

Copyright © Fraunhofer IESE 2007

	• Erweiterte Druckoptionen • Unterstützt Grafikformate wie gif, ps, eps, svg, jpg, und png • Code Generierung für Java • Einfache Installation und updates mit Java Web Start
Erfahrungen	Unterstützt nur 9 Diagramme der UML Es besteht die Möglichkeit je nach kommerzieller Version zusätzliche Funktionalität zu beziehen
Konfigurationstipps	
Bezugsquelle	http://www.gentleware.com

Copyright © Fraunhofer IESE 2007

5 Realisierung

5.1 Integrierte Entwicklungsumgebungen (IDE)

5.1.1 KDevelop

Name und Version	kdevelop v2.1.5
Lizenz / Preis	OpenSource, GPL, frei im Internet verfügbar
Zweck	IDE für C/C++ Projekte
Vorteile	• Integration in KDE Benutzeroberfläche. • Integration des Sourcecodeverwaltungssystems CVS. • Syntax Hervorhebung für verschiedene Dateitypen. • Eingebauter Debugger (basierend auf gdb). • Anzeige und Vervollständigung der Prototypen beim eintippen bekannter Funktionen, Variablen oder Strukturen. • Integration einer Tags Datenbank zum schnellen Suchen bestimmter Funktionen oder Variablen im Projekt. • Integration von doxygen für die Erstellung der eigenen Sourcecodedokumentation. • Applikationstemplates für Konsolen- und grafische Programme mit Unterstützung der autoconf und autobuild tools. • Applikationstemplates für Header- und Quelltextdateien. • Eigener Dokumentationsbrowser mit Suchfunktionalität. • Einbindung der eigenen Dokumentation möglich. • Verschiedene Projektansichten gegliedert nach Verzeichnisstruktur, frei konfigurierbarer logischer Verzeichnisstruktur und Klassen-, Funktions-, Struktur- und Variablenstruktur. • Gute Unterstützung zur Entwicklung grafischer Programme über eingebettete Tools. • Das Programm wird sehr stark weiterentwickelt und neue Funktionalitäten integriert.

Copyright © Fraunhofer IESE 2007

Erfahrungen	• Der integrierte Debugger setzt auf gdb auf und funktioniert gut, wenn das zu untersuchende Programm auf dem gleichen Rechner läuft, auf dem kdevelop auch läuft. • Remote Debugging ist in der Version 2.x nicht möglich. • Programmtemplates können nur sinnvoll genutzt werden, wenn auch die automake und autoconf Tools eingesetzt werden, die die Makefiles automatisch generieren. Andernfalls müssen Makefiles von Hand geschrieben werden. • Doxygen kann über kdevelop sehr schön gestartet und konfiguriert werden. • Sehr große Projekte (z.B. Linux Kernel) sind machbar, der Start von kdevelop dauert dann aber einige Minuten. • Es lohnt sich die neueste Version von kdevelop zu nutzen, da besonders die aktuellen 3.x Versionen sehr viele neue und hilfreiche Features integriert wurden. Zum Beispiel mehr unterstützte Programmiersprachen, Remote Debugging, automake/autoconf Projektmanager, Sourcecodeverwaltungssystem Perforce, Subversion und Clearcase. • Die Anzeige und Vervollständigung von Strukturen und Funktionsdefinitionen funktioniert nicht immer. Zu einigen existierenden Funktionen/Strukturen werden keine Hinweise angezeigt.
Konfigurationstipps	Kdevelop wird mit praktisch jeder aktuellen Linux Distribution ausgeliefert. Zunächst kann die kdevelop Version benutzt werden, die zur KDE Version der Linux Distribution passt. Kdevelop aus den Sourcen zu übersetzen ist nicht ganz einfach und scheitert oft an nicht aktuellen Bibliotheken aus dem KDE Projekt.
Bezugsquelle	http://www.kdevelop.org

5.1.2 Eclipse

Name und Version	Eclipse SDK 3.2
Lizenz / Preis	kostenlos
Zweck	Integrierte Entwicklungsumgebung zur Erstellung von Java-/ und C/C++ Code
Vorteile	• Plattformunabhängige Entwicklungsumgebung (Java-basiert) • Erweiterbarkeit der Plattform durch Plug-ins • Breite Community, daher großes Angebot an Support und Plug-Ins • Integrierte Build-Umgebung

Copyright © Fraunhofer IESE 2007

	• Integrierte Debugging Umgebung • Online Updates der Kernsoftware (Eclipse Plattform) und der jeweiligen Plu-glns • Aktive Unterstützung der Programmierer durch Code Highlighting, Source Navigation, Short cuts, Code completion, Error handling, und vieles mehr.
Erfahrungen	Eclipse wird in Entwicklungen großer Software Projekte bereits erfolgreich einge-setzt.
Konfigurationstipps	
Bezugsquelle	http://www.eclipse.org/

5.1.3 Source Navigator

Name und Version	Source-Navigator v5.1.4
Lizenz / Preis	OpenSource, GPL, frei im Internet verfügbar
Zweck	IDE für C/C++ Projekte
Vorteile	• Syntaxhervorhebung • Integrierter Debugger • Integration von CVS • Darstellung von Funktionen, Variablen und Strukturen in einer durchsuch-baren Baumstruktur. • Aufrufhierarchie für Funktionen darstellbar (statisch). • Headerabhängigkeiten darstellbar.
Erfahrungen	• Aufgrund der sehr guten Querverweise ist ein sehr effizientes arbeiten auch an unbekanntem Code möglich. Die schnelle Navigation im Quelltext er-möglicht es, den fremden Code sehr schnell zu verstehen. • Für sehr große Projekte (Linux Kernel, ca. 12000 Dateien) nicht gut ver-wendbar, da Source Navigator dafür zu langsam ist.
Konfigurationstipps	
Bezugsquelle	http://sourcenav.sourceforge.net/

Copyright © Fraunhofer IESE 2007

5.1.4 Kylix

Name und Version	Kylix 3 Enterprise
Lizenz / Preis	Ca. 2500,-€
Zweck	Pascal-Entwicklungsumgebung (Delphi für Linux)
Vorteile	Da VisionTools V60 komplett unter Delphi entwickelt wurde, ist der Umstieg auf Kylix sehr einfach. Die Entwicklungsumgebung bedient sich auf Anhieb fast wie Delphi in der Windowswelt.
Erfahrungen	Mit Hilfe von Kylix konnten wir recht schnell eine Oberfläche zum Einlesen und Auswerten von Kamerabildern via Firewire realisieren. Es sei jedoch angemerkt, dass Kylix keine runde Pascallösung für Linux ist. Die Entwicklungsumbebung läuft unter WINE und ist keine echte Linuximplementierung. Auch mit verschiedenen Kernelversionen hat Kylix massive Probleme. Der Hersteller unterstützt das Produkt kaum noch und man ist deshalb auf diverse Foren und Tipps von Dritten angewiesen.
Konfigurationstipps	Installation von Kylix unter Suse 9.1: Nach erfolgreicher Installation läuft Kylix nicht (Sanduhr und dann Prozess wieder weg) Lösung: Das Script /usr/local/kylix3/bin/startdelphi muß um folgende Zeile erweitert werden: export LD_ASSUME_KERNEL=2.2.5 Dann läuft die IDE.
Bezugsquelle	Borland

5.1.5 RTAI Lab

Name und Version	RTAI Lab
Lizenz / Preis	GPL / OpenSource
Zweck	Mit RTAI-Lab lassen sich Modelle aus Matlab/Simulink oder Scilab/Scicos automatisch in ausführbaren RTAI-Code umwandeln. Zu diesem Zweck werden spezielle RTAI-blocks Scicos hinzugefügt.

Copyright © Fraunhofer IESE 2007

Vorteile	• Host und target können via net_rpc kommunizieren • Xrtailab stellt ein virtuelles Oszilloskop bereit, mit dem man Daten aus dem Echtzeitprozess visualisieren kann • Automatische Code-Generierung (modell-basierte Entwicklung) • Schnittstelle zu Matlab/Simulink und Realtime-Workshop • Comedi-Support, d.h. Schnittstellen zu Messkarten • Gute Dokumentation mit Beispielen verfügbar
Erfahrungen	-
Konfigurationstipps	-
Bezugsquelle	Teil der RTAI-Distribution (www.rtai.org)

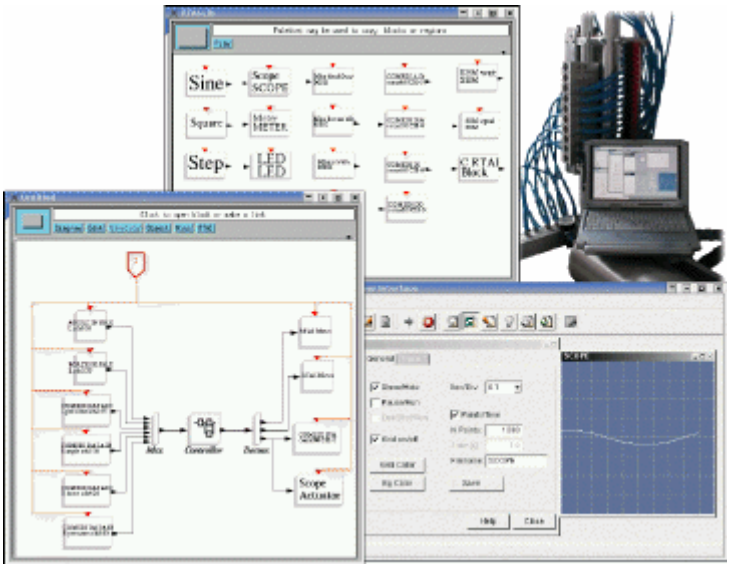


Abbildung 2 RTAI-Lab in Kombination mit Scilab

Copyright © Fraunhofer IESE 2007

5.2 Compiler und Cross-compiler

5.2.1 GNU Compiler Collection (GCC)

Name und Version	gcc 3.2.2 (GNU Compiler Collection)
Lizenz / Preis	OpenSource, GPL, frei im Internet verfügbar
Zweck	Standard-Linux-Compiler für C/C++ und Prozessorarchitekturen.
Vorteile	• Inline Assembler • C und C++ Unterstützung (und weiterer Programmiersprachen) • Strukturelemente von Variablen können elementweise initialisiert werden. Somit können zum Beispiel auch nur einzelne Elemente einer globalen Struktur initialisiert werden. • Unterstützung für Namespaces. • Ist für fast jede Architektur und Betriebssystem zu haben. • Mit ‚gcj' können Java Programme in Native Code übersetzt werden.
Erfahrungen	Version 3.2.2 ist ausgereift, bisher sind in den uns vorliegenden Projekten keine Probleme aufgetaucht. Es lohnt sich nicht, immer die neuste Version zu nutzen, da diese alten Code u.U. anders übersetzt.
Konfigurationstipps	Es lohnt sich meist kaum, gcc selbst zu übersetzen und zu installieren, wenn man keine bestimmte Version benötigt. Gcc ist in allen Linux-Distributionen enthalten.
Bezugsquelle	http://gcc.gnu.org/

5.3 Source Code Dokumentation

5.3.1 Doxygen

Name und Version	doxygen v1.4.3
Lizenz / Preis	OpenSource, GPL, im Internet frei verfügbar
Zweck	Dokumentation des Sourcecode bzw. der API im Projekt

Copyright © Fraunhofer IESE 2007

Vorteile	• Mit den von Doxygen generierten Dokument liegt eine komplette API Beschreibung vor, die automatisch bei jeder Softwarerelease generiert werden kann und dadurch immer aktuell ist. • Verschiedene Ausgabeformate möglich: PDF, HTML, RTF, XML, Latex, Unix ‚man pages'. • Verlinkt die Funktionen, Dateien, Typdeklarationen, Variablendefinitionen etc. so, dass die benötigte Information schnell gefunden wird. • Implementierte Funktionen, Strukturen, Variablen, Typdeklarationen etc. werden automatisch in die Dokumentation aufgenommen. • Code und Dokumentation liegen in der gleichen Datei, dadurch ist die Gefahr gering, dass die zugehörige Dokumentation veraltet. • Grafiken sind in die Ausgabe leicht integrierbar. • Codeteile können in Listen für Beispiele, ToDo Punkte und Bugs aufgenommen werden, die jeweils auf den entsprechenden Codeabschnitt verweisen.
Erfahrungen	• Generiertes Dokumente mit Logos und Layout leicht anpassbar. • Erleichtert und beschleunigt die Dokumentierung des Codes stark. • Sehr ansprechende und umfangreiche Dokumente generierbar, wenn auch speziell bei großen PDF Dokumenten gelegentlich Probleme mit Querverweisen auftreten. • Unterschiedliche Dokumentationsebenen auch über Präprozessoranweisungen realisierbar (z.B. interne Dokumentation und Dokumentation der Kunden API). Treiben den Aufwand für die Dokumentation jedoch in die Höhe und machen die Dokumentation im Quelltext unübersichtlich. • Quelltext wird teilweise aufgebläht. Besonders große Strukturen werden durch viel Dokumentation im Quelltext unübersichtlich. • Auch nicht direkt Sourcecode bezogene Dokumente sind möglich, haben jedoch nur begrenzte Layoutmöglichkeiten und werden schnell unübersichtlich. • Über so genannte Module, die im Inhaltsverzeichnis erscheinen, lässt sich eine übersichtliche Struktur aufbauen. • Von Hand gesetzte Querverweise funktionieren manchmal nicht wie gewünscht. • Tabellen werden speziell in PDF oft durch den Seitenumbruch an ungeschickter Stelle aufgeteilt. • Kommandoabhängigkeiten sind in der Dokumentation zu doxygen nicht immer ersichtlich (Ein \section Kommando funktioniert nur in einer \page Umgebung korrekt). • Nicht alle Formatierungsanweisungen werden in allen Ausgabeformaten

Copyright © Fraunhofer IESE 2007

	wie erwartet übernommen. • Bestimmte Layoutvorstellungen sind manchmal nicht oder nur mit großem Aufwand realisierbar. • Der optionale Treeview arbeitet mit gängigen Browsern (Mozilla/Opera/IE) nicht immer korrekt zusammen. • Bei hohen Layoutansprüchen muss viel Zeit für die saubere Darstellung der Dokumentation investiert werden.
Konfigurationstipps	• Sollen zwei verschiedene Dokumente aus den gleichen Quelltexten erstellt werden, ist es sinnvoll, Elemente für die verschiedenen Dokumente nicht in einer Datei zu vermischen. Damit werden unübersichtliche Präprozessoran-weisungen vermieden. • Am Besten funktioniert die Generierung der HTML Dokumentation. Große PDF Dokumente sind problemloser über den Umweg Latex erstellen als über doxygen direkt.
Bezugsquelle	http://www.doxygen.org

5.4 Debugger

5.4.1 GNU Debugger

Name und Version	gdb (GNU Debugger)
Lizenz / Preis	OpenSource, GPL, frei im Internet verfügbar
Zweck	Standard-Linux-Debugger der GNU Toolchain für Userspace Programme
Vorteile	• ‚Standarddebugger' für Linux Systeme. • Remote Debugging mit Hilfe des gdbserver Programms auf dem Zielsystem • Debugging von Threads
Erfahrungen	• Hoher Einarbeitungsaufwand für Textbasierte Kommandooberfläche • Zeigt nur Standard CPU Register an. Keine Unterstützung der erweiterten Register von embedded Controllern, der MMU- oder Cache Einheit. • Das Remote Debugging mit gdbserver Programm funktioniert sehr gut.
Konfigurationstipps	Um Remote Debugging von Applikationen zu ermöglichen wird nur das Pro-gramm gdbserver auf dem Zielsystem benötigt. Das gdbserver Programm ist so klein, dass es auch auf kleinen embedded Systemen auf dem Zielsystem abgelegt

Copyright © Fraunhofer IESE 2007

	werden kann. Die Programme, die mit gdbserver gedebuggt werden sollen, brauchen auf dem Zielsystem keine Debuginformationen enthalten. Nur auf dem Rechner, auf dem die Debugoberfläche (z.B. gdb) läuft müssen dann die kompilierten Programme und Bibliotheken mit Debuginformationen vorliegen. Gdb und gddbserver müssen auch Cross-Kompiliert werden und zum jeweiligen Zielsystem passen. Für das Remote Debugging muss das gdbserver Programm mit den Kommunikationsparametern und dem zu debuggenden Programm ausgeführt werden. Dafür sollte ein Zugang zum Zielsystem zur Verfügung stehen mit dem das gdbserver Programm gestartet werden kann (z. B. Telnet, Konsole oder ssh).
Bezugsquelle	http://www.gnu.org/software/gdb/gdb.html

5.4.2 Data Display Debugger (ddd)

Name und Version	ddd (Data Display Debugger)
Lizenz / Preis	OpenSource, GPL, frei im Internet verfügbar
Zweck	Grafischer Aufsatz auf gdb Kommandozeilenprogramm
Vorteile	• Zeigt Debugumgebung des gdb grafisch übersichtlich an • Zeigerstrukturen und verkettete Listen werden grafisch übersichtlich dargestellt • Remote Debugging fähig • Einfache Bedienung, weiterhin Eingabe der Befehle über Textkonsole möglich. • Assembler und Sourcecode gleichzeitig darstellbar
Erfahrungen	Stabiles System Bedienung einfach, ähnlich mit vergleichbaren Debuggern Funktionalität und benötigte Schritte für Remote Debugging teilweise verwirrend. Wegen fehlender Syntaxhervorhebung leidet Übersichtlichkeit des Sourcecodes
Konfigurationstipps	Über Startdateien kann die benötigte Konfiguration automatisch eingestellt werden.

Copyright © Fraunhofer IESE 2007

Bezugsquelle	http://www.gnu.org/software/ddd/

5.4.3 Lauterbach

Name und Version	Lauterbach
Lizenz / Preis	Proprietär / ca. 3000€ (JTAG Hardware und Software) je nach eingesetzter Hardware
Zweck	Hardware Debugger für versch. Architekturen; Debugschnittstelle JTAG; Debugger für Kernel und Kerneltreiber
Vorteile	• Volle Kontrolle über die gesamte Hardware • Debuggen im Kernel, Treibern und von Userspace Applikationen möglich • Bringt eigene Debugoberfläche mit, die sehr übersichtlich gestaltet und speziell auf embedded Systeme und Hardwarenahe Programmierung bzw. Hardwareinbetriebnahmen ausgelegt ist. • Auch Spezielle Register der verwendeten Hardware werden grafisch übersichtlich angezeigt. • Debugger erweiterbar mit Trace, Probe, etc. Einheiten für erweitertes Hardwaredebuggen und Inbetriebnahmen. • Debugoberfläche läuft auf Windows oder Linux. • Kann Flashbausteine direkt programmieren • Umfangreiche Skriptingmöglichkeiten • Einzige Möglichkeit Linux Kernel- oder Treibercode zu debuggen ohne Patches auf den Kernel anzuwenden.
Erfahrungen	Debugging von Kernel- / Treibercode ist heikel und schwierig Debugging von Applikationscode noch schwieriger als Kernelcode. Der Einsatz erfordert genaue Kenntnisse des Linux Kernels
Konfigurationstipps	
Bezugsquelle	http://www.lauterbach.de

Copyright © Fraunhofer IESE 2007

5.4.4 BDI 2000

Name und Version	BDI 2000
Lizenz / Preis	Proprietär / ca. 2500€ (JTAG Hardware und Software) je nach eingesetzter Hardware
Zweck	Hardware Debugger für versch. Architekturen; Debugschnittstelle JTAG; Debugger für Kernel und Kerneltreiber
Vorteile	• Volle Kontrolle über die Hardware • Debuggen im Kernel und Treibern möglich • Implementiert das gdbserver Protokoll. Damit können alle grafischen Oberflächen benutzt werden, die Remotedebugging mit dem gdb erlauben (z.B. ddd) • Erweiterte Kommandos des gdb für Hardwarebreakpoints etc. • Kann Flashbausteine direkt programmieren • Einzige Möglichkeit Linux Kernel- oder Treibercode zu debuggen ohne Patches auf den Kernel anzuwenden.
Erfahrungen	Debugging von Kernel- / Treibercode ist heikel und schwierig Der Einsatz erfordert genaue Kenntnisse des Linux Kernels
Konfigurationstipps	
Bezugsquelle	http://www.abatron.ch

Copyright © Fraunhofer IESE 2007

6 Systemintegration

Dieses Kapitel umfasst Werkzeuge zur Systemintegration und zur Verwaltung von Releases (Release-Management).

6.1 Build-Tools

6.1.1 Ant und Anthill

Name und Version	**Ant 1.6.5** http://www.urbancode.com/projects/anthill/
Lizenz / Preis	kostenlos
Zweck	Build Werkzeug, Build Prozess Management
Vorteile	• **Ant** • Plattformunabhängig (vgl. Make) • offenes System mit exakt definierten Schnittstellen • Viele Java-Werkzeuge bieten Unterstützung für Ant • Kann im Anthill benutzt werden • Ist in Eclipse bereits integriert • **Anthill** • führt eine Prüfung vom Repository der neuesten Version eines Projektes vor jedem Build durch und etikettiert den Behälter mit einer einzigartigen Nummer nach jedem Build • Unterstützt eine Vielzahl von Respositories: CVS (Concurrent Versions System), Clearcase, MKSIntegrity, Perforce, PVCS, StarTeam, Subversion Visual Source Safe, und FileSystem. • Aktualisiert automatisch die Projekt Intranet Seite des letzten Builds • Jeder Build wird aufgezeichnet • Jeder Build kann wiederhergestellt werden
Erfahrungen	

Copyright © Fraunhofer IESE 2007

Konfigurationstipps	
Bezugsquelle	http://ant.apache.org/ http://www.urbancode.com/projects/anthill/

6.1.2 GNU Make

Name und Version	Make V3.81
Lizenz / Preis	GPL/Open Source
Zweck	Bedingtes Kompilieren/Linken von Quellkode (Hauptzweck). In einem sogenannten „Makefile“ werden Ziele definiert und Regeln, wie das Ziel zu erreichen ist. Make prüft die Regeln und erstellt die Ziele. So können beispielsweise mit einem einfachen Aufruf Dateien kompiliert werden, Testläufe gestartet werden oder Dokumentationen generiert werden. Make stellt dabei sicher, dass z.B. Abhängigkeiten zwischen Dateien berücksichtigt werden.
Vorteile	• Standard für C/C++-Programmierung • Sehr flexibel und kompakte Darstellung
Erfahrungen	GNU Make unterstützt deutlich mehr Funktionen als das „Standard“ Unix-Make, ist jedoch auch schwierig zu erlernen
Konfigurationstipps	Anhand von Makefiles anderer Open Source Projekte kann man viel über Konfiguration und Arbeitsweise lernen. Oft führt die Anpassung eines solchen Makefiles schnell zum Ziel.
Bezugsquelle	http://www.gnu.org/software/make/

Copyright © Fraunhofer IESE 2007

7 Qualitätssicherung

7.1 Testing Tools

7.1.1 Gcov Coverage Tool

Name und Version	Gcov, Version 1.5
Lizenz / Preis	Open Source
Zweck	Profiling Werkzeug zur Performanceanalyse des Codes
Vorteile	• Ist im gcc Framework enthalten, so dass nur ein geringer Konfigurationsaufwand besteht • Ist erweiterbar für Echtzeitaspekte • Liefert Informationen zur Ausführungshäufigkeit jeder einzelnen Codezeile • Liefert Informationen zur Bearbeitungszeit jeder Codesektion
Erfahrungen	Die Ausgabe des Programms zeigt den Code und die Häufigkeit der Ausführung jeder einzelnen Zeile. Die Anzeige ist sehr schlicht (Zeilennummer, Durchführungshäufigkeit, Codezeile), bietet jedoch alle nötigen Informationen. Weiterhin ist es möglich, sich die Zweigüberdeckung anzeigen zu lassen. Die Erweiterung von gcov, mit der Dateien, die im Kernel Space ablaufen, analysiert werden können, hat sich durch die Installation unterschiedlicher Patches als schwierig erwiesen. Für den Demonstratorcode wurde das Werkzeug eingesetzt. Dabei konnten häufig benutzte Funktionen sehr gut durch die extrem hohe Wiederholung von Zeilendurchführungen identifiziert werden. Codefragmente, die nicht durchlaufen wurden, konnten so kritisch hinsichtlich ihrer Wichtigkeit analysiert werden (z.B., Handelte es sich um einen else Block oder um „toten“ Code).
Konfigurationstipps	Gcov wird durch eine Parameterangabe bei der Kompilierung mittels gcc durchgeführt. Auf der Ausgabedatei wird dann gcov angewendet. Ein detailliertes Beispiel ist unter der angegebenen Bezugsquelle zu finden.
Bezugsquelle	http://gcc.gnu.org/onlinedocs/gcc-3.0/gcc_8.html

Copyright © Fraunhofer IESE 2007

7.1.2 CppUnit

Name und Version	CppUnit 1.10.2
Lizenz / Preis	Open Source
Zweck	Unit Testing Framework für C++
Vorteile	• Leicht zu benutzen • Basiert auf dem bekannten JUnit Framework • Kann in Verbindung mit gcc und VC++ benutzt werden • Durch Timer Komponenten auch für Echtzeitbereich einsetzbar
Erfahrungen	Benutzerfreundlich Leicht in die integrierte Entwicklungsumgebung einzubetten
Bezugsquelle	http://cppunit.sourceforge.net/cppunit-wiki

7.2 Analyse-Werkzeuge

7.2.1 Linux Trace Toolkit

Name und Version	LTT (Linux-Trace-Toolkit)
Lizenz / Preis	OpenSource, GPL, frei im Internet verfügbar
Zweck	Grafische und Text basierte zeitbezogene Darstellung von Tasks (Threads)
Vorteile	• Zeigt graphisch Prozesse und Laufzeiten an • Auflösung im µs-Raster • Darstellung Linux- und RTAI-Events • Anzeige Ressourcenverbrauch pro Task
Erfahrungen	Stabiles System Bedienung einfach Sehr träges Fensterverhalten.
Bezugsquelle	http://www.opersys.com

Copyright © Fraunhofer IESE 2007

8 Beispiele für Toolketten

In diesem Kapitel werden Beispiele konkreter Toolketten gegeben, wie sie bei den Projektpartnern im realen Einsatz sind.

8.1 Entwicklung eingebetteter Linux Realtime-Systeme (Berghof)

BERGHOF hat bei der Auswahl der Toolkette den Focus auf eingebettete Systeme gerichtet. Dabei ist es wichtig, die Anwendungsszenarien im Auge zu behalten. Die Toolkette muss einerseits in der Büroumgebung einer Entwicklungsabteilung in Kombination mit der Windows-dominierten Officewelt funktionieren, andererseits aber auch vor Ort an Maschinen und Anlagen im Feld einsetzbar sein. Deshalb ist neben den reinen Tools auch die Hard- und Software-Infrastruktur mit zu berücksichtigen.

Entlang der im Entwicklungsprozess notwendigen Aufgaben wurden Tools gesichtet, erprobt und zu einer funktionierenden Kette zusammengefügt. Diese Zusammenstellung darf nicht als die verbindliche, sondern nur als eine mögliche betrachtet werden. Sie soll dem Neueinsteiger helfen, schnell zu einer funktionierenden Umgebung zu kommen, aus der er selbst seine individuelle Werkzeugkette zusammenstellen und weiter optimieren kann. Der ansonsten nötige Einarbeitungsaufwand mit Sichtung, Erprobung und Auswahl soll entfallen.

Für die einzelnen Phasen des Entwicklungsprozesses wurden folgende Tools ausgewählt:

- **Spezifikation / Design:**
  Kommerzielle Produkte: Microsoft Word, Microsoft Visio, Mindmanager
  Open-Source-Produkte: Open Office, Freemind
  Anmerkung: Im Rahmen der Spezifikation wird eng mit dem Kunden kommuniziert, der oft nur über die Microsoft-Office-Produkte verfügt. Deshalb bietet sich hier noch an, ohnehin vorhandene kommerzielle Produkte zu verwenden.
- **Entwicklung:**
  Open-Source-Produkte:
  Integrierte Entwicklungsumgebung IDE Kdevelop mit Editor
  Compiler GCC für PowerPC
- **Fehlersuche:**
  Debugger GDB für PowerPC
  HW-Debugger Lauterbach (HW-bezogen)

Copyright © Fraunhofer IESE 2007

- **Versionsverwaltung:**
  Subversion
- **Dokumentation:**
  Doxygen
- **Verbesserung / Fehlertracking:**
  Mantis

Ausgegangen wurde von der vorliegenden IT-Infrastruktur und der Notwendigkeit, bei Kundenprojekten sowohl stationäre als auch mobile Lösungen zur Verfügung haben zu müssen (z. B. Labtop für das Nachprogrammieren vor Ort).

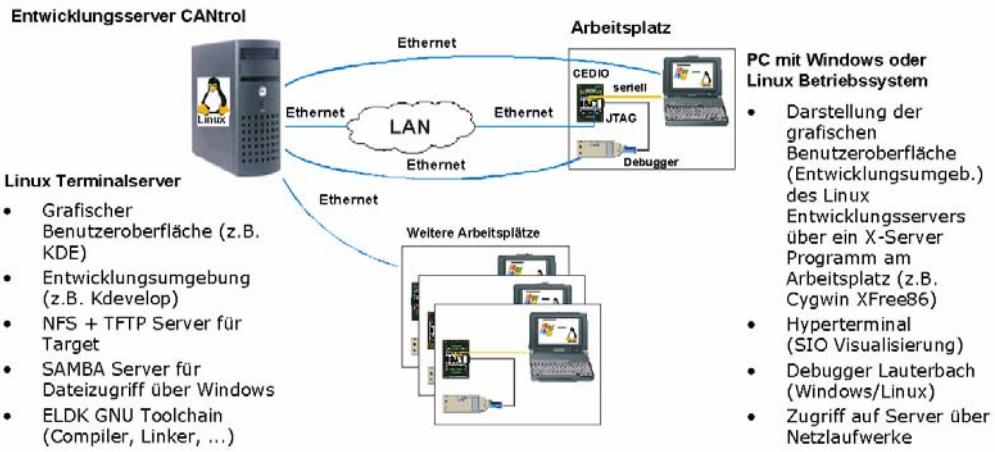


Abbildung 3: Infrastruktur: Entwicklungsabteilung mit Server

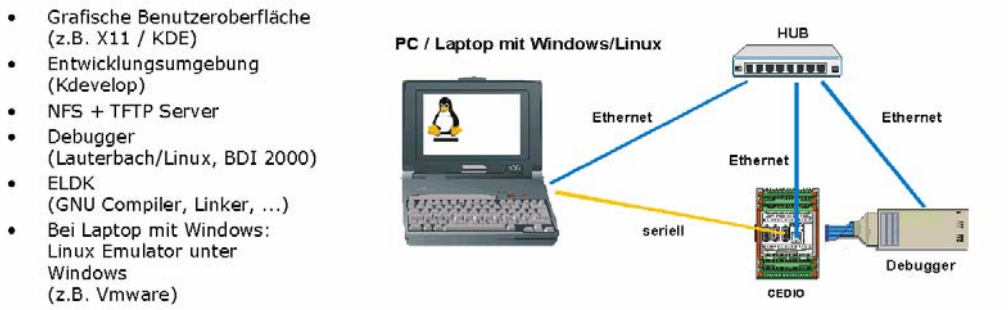


Abbildung 4: Mobiler Einsatz / Stand alone

Hiervon ausgehend wurden für die unterschiedlichen Phasen des Entwicklungsprozesses sukzessive Tools untersucht, ausgewählt und in Betrieb genommen.

Copyright © Fraunhofer IESE 2007

Vordergründiges Kriterium dabei war – dem allgemeinen Grundgedanken von RTLOpen folgend – auch für die notwendigen Entwicklungstools soweit als möglich auf Open-Source-Produkte zurückzugreifen.

Im Zusammenhang mit diesen Arbeiten wurden neben der grundsätzlichen Nutzung von Open-Source-Software noch zwei weitere wichtige Kriterien für die Qualitätsbeurteilung von Tools herausgearbeitet. Es sind dies:

- Eine leichte Einarbeitung in das Tool, d. h. das Tool muss schnell einsatzfähig sein und geringe Einarbeitungskosten erfordern.
- Die Tools müssen insgesamt gut aufeinander abgestimmt sein, damit ein schneller Arbeitsablauf garantiert ist und Doppeleingaben oder Konvertierungen unterbunden werden.

Mit den so festgelegten Rahmenbedingungen wurde nun der gesamte Entwicklungsprozess zuerst grundsätzlich bezüglich benötigter Toolfunktionen und im weiteren Verlauf dann in Bezug auf konkrete Tools durchleuchtet. Die nachfolgenden beiden Abbildungen verdeutlichen dies.

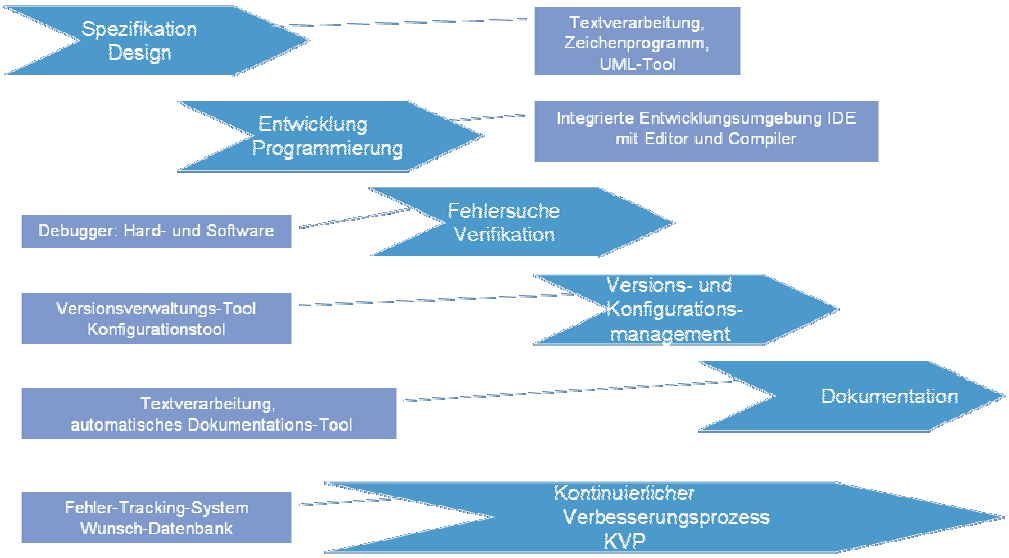


Abbildung 5: Grundsätzliche Funktionalitäten der Toolkette entlang des Entwicklungsprozesses

Copyright © Fraunhofer IESE 2007

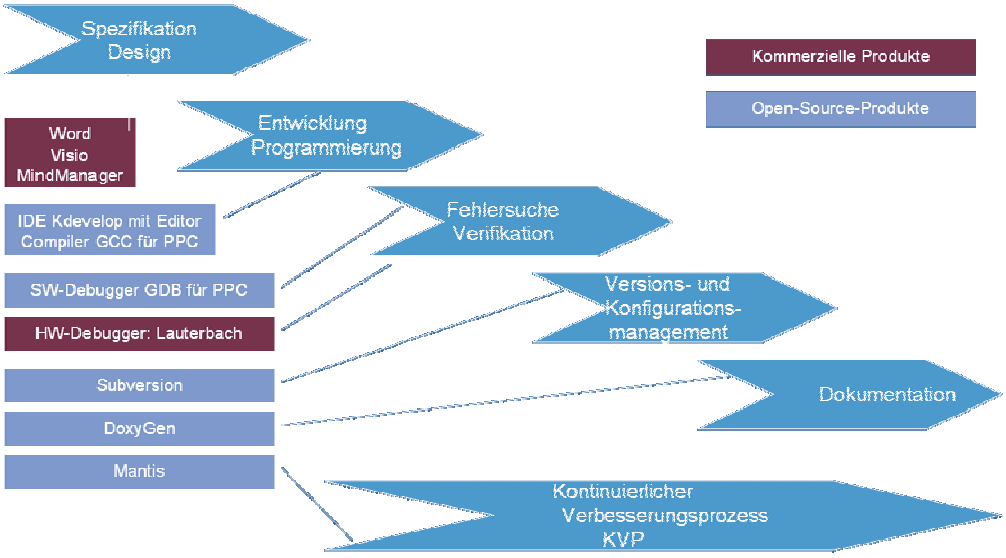


Abbildung 6: Auswahl konkreter Tools entlang des Entwicklungsprozesses bei Berghof

Wie man sieht, ist es tatsächlich gelungen, die Mehrzahl der Funktionen mit Open-Source-Produkten zu belegen. Berücksichtigt man weiterhin, dass im Grunde die für die Spezifikations- und Designphase notwendigen Tools in den meisten Office-Anwendungen ohnehin enthalten sind, reduziert sich die Zahl der verwendeten Spezialtools im Grunde auf „1“. Hierbei handelt es sich um den Hardware-Debugger, bei dem es in der Natur der Sache liegt, dass dieser wohl auch in Zukunft individuell bleiben wird.

Zusammenfassend lässt sich sagen, dass der durch Linux getriebene Open-Source-Gedanke Produkte hervorgebracht hat, die sich im industriellen Umfeld einsetzen lassen. Es gibt Hürden bei der Einführung, die zu bewältigen sind. Neben der Auswahl der richtigen Tools aus der großen verfügbaren Menge macht ihre Erstinstallation mit Zuteilung der richtigen Userrechte oft Probleme. Danach laufen die meisten etablierten Tools stabil und treffen auf große Akzeptanz bei den Entwicklern. Durch die kostenfreie Verfügbarkeit ist sogar die Bereitschaft höher, auch für ungeliebtere Tätigkeiten wie z.B. das Dokumentieren und Versionieren Tools einzusetzen und dann auch konsequent zu benutzen.

8.2 Entwicklung eines RTAI Subsystems (Hofmann)

Für die Entwicklung des RGM – Messkerns wurden die folgenden Tools und Entwicklungsumgebungen benutzt:

- RTAI 24.1.9
- SuSE Linux-Distribution V. 7.0

Copyright © Fraunhofer IESE 2007

- GNU-Entwicklungstools (make V. 3.79.1, gcc-Compiler V. 2.95.2)
- Graphische Benutzeroberfläche KDE V. 1.1.2
- Graphische integrierte Entwicklungsumgebung KDevelop V 1.2

Auswahl der Linux-Distribution

Bei der Auswahl der Linux-Distribution wurden die Distributionen von SuSe und Red Hat untersucht. Zum damaligen Zeitpunkt waren dies die am meisten verbreiteten Linux- Distributionen in Deutschland, was auch heute im Wesentlichen noch gilt.

Technisch gesehen ähneln sich die Distributionen weitgehend und können durch nachträgliche Konfiguration so konfiguriert werden, so dass gravierende funktionale Unterschiede nicht ausgemacht werden können.

Daher wurden Kriterien wie

- Einfache Installation, Wartung und Bedienung
- Verfügbarkeit von Dienstleistungen und Support
- Marktposition

zur Entscheidungsfindung herangezogen.

Während der Einsatzort von Red Hat überwiegend im Serverbereich liegt, hat sich SuSe im Desktop Bereich etabliert. Die Gründe dafür sind die einfache Installation und Bedienung des Betriebssystems. Dies wurde durch eine weitgehend automatisierte Installationsroutine realisiert. Das integrierte grafische Userinterface KDE ermöglicht einen schnellen Umstieg von der Windows- in die Linux-Welt durch die weitgehende gleiche Bedienung. Ebenso ist die Verfügbarkeit von Dienstleitungen und professionellen Support gewährleistet. Zudem wird eine Vielzahl von Hardwareplattformen unterstützt. Neben einer starken lokalen Marktposition besitzt SuSe noch den strategischen Vorteil mit dem Firmensitz in Deutschland.

Zur Entwicklung der Anwendungen wird bei Hofmann noch die Programmiersprache C eingesetzt. Dies ist historisch bedingt und hat zudem den Vorteile, dass man die Anwendungen leicht auf andere Hardware portieren kann, sofern ein C-Compiler für diese existiert.

Mit den GNU-Entwicklungstools, der grafischen Bedieneroberfläche KDE und KDevelop liefert die Suse-Distribution alle zur Entwicklung benötigen Werkzeuge gleich mit.

Copyright © Fraunhofer IESE 2007

8.3 Linux Anwendungsentwicklung mit Realzeit-Komponenten (VisionTools)

VisionTools-V60 ist komplett mit Borland Delphi realisiert worden. Die zugrundeliegende Sprache ist Pascal. Um mit etwas Weitblick eine wirtschaftliche Entwicklungsumgebung unter Linux auszuwählen, ist die einfache Portierbarkeit der Algorithmen ein wesentliches Kriterium. Das die Windowswelt auch weiterhin aktuell bleiben wird, ist die parallele Wartung und Erweiterbarkeit in beiden Welten (Linux und Windows) notwendig. Hierbei ist Delphi für VisionTools die optimale Lösung. In Verbindung mit Kylix kann hier eine Flexibilität und Wirtschaftlichkeit erreicht werden, die kein Entwicklungswerkzeug der C zugrunde liegt leisten kann. Die für uns besonderen Leistungsmerkmale soll das folgende Schaubild verdeutlichen:

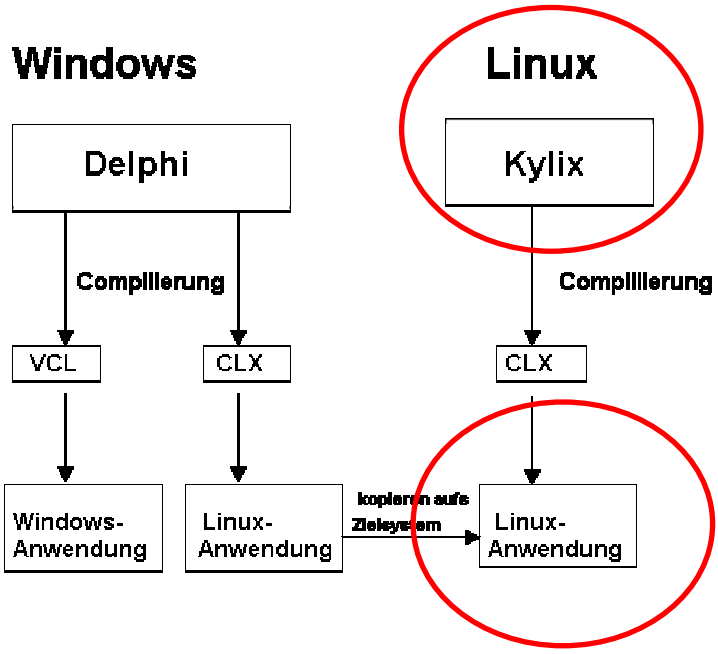


Abbildung 8-7 Migration von Windows Delphi nach Linux mit Kylix bei VisionTools

Copyright © Fraunhofer IESE 2007

Werkzeug	**Funktion**	**Benutzte Version**	**Bezugsquelle**
Suse 9.1 professional	Linux-Distribution mit u.a.: Kernel 2.6.8 KDE 3.2 Kdevelop 3.0 GNOME 2.4 Samba 3.0 YaST2 OpenOffice	V9.1	Softwarehandel
RTAI	Echzeit-Kernelpatch für Linux	V3.1	http://rtai.org
Libraw1394	Libraw1394 ist eine einheitliche Schnittstelle zum Firewirebus und stellt den direkten Zugriff auf ihn zur Verfügung.	V1.1.0	http://sourceforge.net/projects/libraw1394
Libdv	Libdv decodiert die Firewire-Daten entsprechend dem DV-Videoformat (IEC 61834 und SMPTE 314M). Dieses Videoformat findet man bei fast allen gängigen Videokameras vor allem im Consumer-Bereich. Im Industriebereich gibt es noch den IIDC-Standard der hier zunächst unberücksichtigt bleibt.	V0.104	http://sourceforge.net/projects/libdv
Coriander	Mit Coriander können Bilder über den Firewirebus eingelesen werden und wahlweise zum Display oder in Dateien geschrieben werden. Wir haben hier, wie oben erwähnt, das Device /dev/video1394/0 gewählt (Mode Video1394 nicht RAW1394 einstellen).	V1.0.1	http://sourceforge.net/projects/coriander
Borland Kylix	Delphi-Entwicklungsumgebung unter Linux.	V3.0 Enterprise	http://www.borland.com
Qt	Entwicklungsumgebung unter Linux für KDE	V3.3	http://www.trolltech

Copyright © Fraunhofer IESE 2007

Dokumenten Information

Titel:	RTLOpen Plattform Handbuch Teil 5 – Entwicklungswerkzeuge
Datum:	1. September 2007
Report:	IESE-057.07/D
Status:	Final
Klassifikation:	Öffentlich

Copyright 2007, Fraunhofer IESE.
Alle Rechte vorbehalten. Diese Veröffentlichung darf für kommerzielle Zwecke ohne vorherige schriftliche Erlaubnis des Herausgebers in keiner Weise, auch nicht auszugsweise, insbesondere elektronisch oder mechanisch, als Fotokopie oder als Aufnahme oder sonstwie vervielfältigt, gespeichert oder übertragen werden. Eine schriftliche Genehmigung ist nicht erforderlich für die Vervielfältigung oder Verteilung der Veröffentlichung von bzw. an Personen zu privaten Zwecken.