Projekt Exquisite

Kurzdokumentation



März 2013

**Inhaltsverzeichnis**

[Vorwort 1](#_Toc350786194)

[Copyright / Lizenz: 1](#_Toc350786195)

[Teil I: Einführung in Exquisite 2](#_Toc350786196)

[1 Einleitung 3](#_Toc350786197)

[1.1 Hintergrund 3](#_Toc350786198)

[1.2 Ziele 3](#_Toc350786199)

[2 Einführung in die Constraint-Programmierung 5](#_Toc350786200)

[2.1 Constraints, Constraint-Systeme und Constraint-Solver 5](#_Toc350786201)

[2.2 Choco Constraint-Solver 5](#_Toc350786202)

[3 Einführung in die Excel 2010-Programmierung mit VSTO und .NET 4.0 7](#_Toc350786203)

[3.1 Überblick VSTO 7](#_Toc350786204)

[3.2 Das Objektmodell von Excel 7](#_Toc350786205)

[3.3 Interaktion mit anderen Technologien 8](#_Toc350786206)

[3.3.1 Das neue Dateiformat von Office (Excel) im Überblick 9](#_Toc350786207)

[3.3.2 Custom XML Parts 9](#_Toc350786208)

[4 Einführung in Exquisite 11](#_Toc350786209)

[4.1 Das Excel-Add-In (Client) 11](#_Toc350786210)

[4.2 Der Exquisite-Server 11](#_Toc350786211)

[4.3 Die Datenhaltung und Kommunikation 11](#_Toc350786212)

[Teil II: Exquisite Systemstruktur 12](#_Toc350786213)

[5 Konzeption und Vorüberlegungen 13](#_Toc350786214)

[5.1 Das Client-Server-Modell 13](#_Toc350786215)

[5.2 XML als Schnittstelle und neutrales Sicherungsformat 13](#_Toc350786216)

[5.3 Choco als Constraint-Solver 13](#_Toc350786217)

[6 Der Client 15](#_Toc350786218)

[6.1 Systemanforderungen und Voraussetzungen 15](#_Toc350786219)

[6.2 Werkzeuge 16](#_Toc350786220)

[6.3 Die Installation 16](#_Toc350786221)

[6.3.1 Installationsdateien überprüfen 16](#_Toc350786222)

[6.3.2 Der Installationsverlauf 17](#_Toc350786223)

[6.4 Konfiguration 18](#_Toc350786224)

[7 Der Server 23](#_Toc350786225)

[7.1 Systemanforderungen und Voraussetzungen 23](#_Toc350786226)

[7.2 Werkzeuge 23](#_Toc350786227)

[7.3 Installation 23](#_Toc350786228)

[7.3.1 Installationsdateien überprüfen 23](#_Toc350786229)

[7.3.2 Der Installationsverlauf 23](#_Toc350786230)

[7.4 Konfiguration 23](#_Toc350786231)

[Teil III: Exquisite als integriertes Werkzeug zur Fehlersuche 24](#_Toc350786232)

[8 Der erste Programmstart 25](#_Toc350786233)

[8.1 Der Debug-Mode 25](#_Toc350786234)

[9 Die Benutzerschnittstelle 27](#_Toc350786235)

[9.1 Exquisite-Menüband (Ribbon) 27](#_Toc350786236)

[9.1.1 Testfälle 27](#_Toc350786237)

[9.1.2 Diagnose 27](#_Toc350786238)

[9.1.3 Konfigurationsmöglichkeiten 27](#_Toc350786239)

[9.1.4 Werkzeuge (Utilities) 27](#_Toc350786240)

[9.2 Exquisite-Steuerelemente (UserControls) 27](#_Toc350786241)

[9.2.1 Testfallmodellierung 27](#_Toc350786242)

[9.2.2 Debug-Fenster und visuelle Feedbacks 28](#_Toc350786243)

[10 Diagnose 29](#_Toc350786244)

[Teil IV: Schlussbemerkungen 30](#_Toc350786245)

[11 Häufige Fehler 31](#_Toc350786246)

[12 Häufig gestellte Fragen 33](#_Toc350786247)

[Anhang 36](#_Toc350786248)

[A. Entwicklungsumgebung I](#_Toc350786249)

[B. Verzeichnisstruktur des Servers II](#_Toc350786250)

[C. Verzeichnisstruktur des Clients III](#_Toc350786251)

[Glossar IV](#_Toc350786252)

[Literaturverzeichnis V](#_Toc350786253)

[Abbildungsverzeichnis VI](#_Toc350786254)

[Index VII](#_Toc350786255)

# Vorwort

**Hinweise zur vorliegenden Kurzdokumentation**:

Datum der letzten Bearbeitung: 19.03.2013

zuletzt bearbeitet durch: Arash Baharloo

**Dieses Handbuch ist einer laufenden Bearbeitung unterworfen**

# Copyright / Lizenz:

Teil I:  
Einführung in Exquisite

# Einleitung

## Hintergrund

## Ziele

# Einführung in die Constraint-Programmierung

## Constraints, Constraint-Systeme und Constraint-Solver

## Choco Constraint-Solver

# Einführung in die Excel 2010-Programmierung mit VSTO und .NET 4.0

Der Vorteil Microsoft Excel als Front-End zu verwenden besteht darin, dass man die vertrauten Microsoft Office-Benutzeroberflächen und Tools z.B. die Datenanalysefunktionen nutzen kann. Es existieren viele Möglichkeiten Office-Anwendungen zu erweitern[[1]](#footnote-1).

Hinweis: weitere Informationen zur Office-Entwicklung finden Sie auf der offiziellen Webseite (siehe [1]).

Es werden immer noch die meisten auf Microsoft Office basierende Anwendungen ohne Visual Studio Tools für Office (VSTO) umgesetzt. Dabei bietet VSTO die Möglichkeit, Softwarelösungen für Office-Anwendungen deutlich schneller, eleganter und bei manchen Aufgaben sogar effizienter zu entwickeln.

VSTO bietet nicht nur einen einfachen Einstieg in die Office Programmierung, sondern auch die volle Unterstützung des .NET Frameworks.

## Überblick VSTO

Visual Studio Tools für Office (VSTO) ist im Wesentlichen eine Menge von Projektvorlagen für die Entwicklungsumgebung Visual Studio, die von Microsoft entwickelt und bereitgestellt wird. Mit diesen Vorlagen ist es möglich, Lösungen für Office-Anwendungen unter Verwendung der .NET Programmiersprachen (Visual Basic oder Visual C#) zu entwickeln. Da so entwickelte Lösungen auf .NET Framework basieren, verbinden sie die Office-Anwendungen mit .NET Framework im vollen Umfang.

## Das Objektmodell von Excel

Die Objektmodelle der Office Anwendungen basieren auf COM. Das Objektmodell ist dabei die Art und Weise, wie man mit dem jeweiligen Programm umgehen kann, und auf die einzelnen Objekte, die es dort gibt, zugreifen kann.

Ein Beispiel: Man erzeugt sich ein Objekt vom Typ „Excel.Application“, und hat damit einen Zeiger auf das bereits laufende Excel-Programm. Dann ist es möglich mit Befehlen wie „Worksheets.Add“ einfach ein neues Arbeitsblatt erzeugen, so als ob man in Excel „Datei / Neu“ aufrufen würde.

Um aus einer .NET Anwendung auf diese COM-Komponenten zugreifen zu können, werden besondere Wrapperklassen, s.g. Interop-Assemblies, benötigt. Wenn ein neues Office-Projekt in Visual Studio erstellt wird, fügt Visual Studio automatisch Verweise auf die primären Interopassemblys (PIAs) von Microsoft Office hinzu, die zum Erstellen des Projekts erforderlich sind. Die PIAs werden in vorkompilierter Form von Microsoft verteilt.

Die PIAs beinhalten eine Menge von signierten Wrapperklassen und ermöglichen verwaltetem Code, mit dem COM-basierten Objektmodell einer Microsoft Office-Anwendung zu interagieren.[[2]](#footnote-2)

Abbildung 3‑1: Ausschnitt aus dem Excel-Objektmodell

Das Verständnis, wie Excel intern arbeitet, ist eine besonders wichtige Grundlage für die Programmierung einer Excel-basierten Anwendung. Abbildung 3‑1 Zeigt einen Ausschnitt aus dem Objektmodell von Excel. Mehr Informationen zum Excel-Objektmodell findet sichin der offiziellen Dokumentation [2].

Hinweis: Es existieren in Excel keine Cell-Objekte für die einzelnen Zellen in einem Arbeitsblatt, sondern Range-Objekte, die sowohl eine einzelne Zelle als auch einen Bereich von mehreren Zellen repräsentieren.

## Interaktion mit anderen Technologien

Der Einstieg in die OfficeEntwicklung erfolgt aufgrund des kostenlosen Vorhandenseins der zugehörigen Entwicklungsumgebung in der Regel mit Visual Basic für Applikationen (VBA). VBA verwendet nicht verwalteten Code, der in die Office Anwendungen integriert ist und bietet eine einfache Möglichkeit einfache Dokumentfunktionen zu entwickeln und mit Hilfe des Makrorekorders einfach Makros aufzuzeichnen.

Beim parallelen Einsatz verschiedener Techniken und Technologien ist meistens eine reibungslose Interaktion erforderlich. In einer Excel-Umgebung ist es problemlos möglich VBA-Code aus .NET aufzurufen und umgekehrt.

Um VBACode aus .NET aufzurufen, muss lediglich die Methode „Run“ des Application-Objekts, das die Excel-Anwendung repräsentiert, aufgerufen werden. Für den Aufruf von .NETCode aus VBA ist in der „ThisAddIn“-Klasse die Methode „RequestComAddInAutomationService“ so zu überschreiben, dass sie ein Objekt der Klasse zurückgibt, die nach außen verfügbar gemacht werden soll. Die Klasse selbst muss dafür über das „ComVisible“-Attribut verfügen, das auf „true“ festgelegt ist. Die VSTORuntime weist das zurückgegebene Objekt der ObjectEigenschaft eines COMAddInObjekts in VBA zu, wodurch dieses nach außen zur Verfügung steht[[3]](#footnote-3).

### Das neue Dateiformat von Office (Excel) im Überblick

Die neuen Dateiformate ab Office 2007 basieren auf der „Office Open XML Document Interchange Specification“. Zu erkennen sind sie an der Dateiendung: .xlsx (Excel).

Bei diesen Dateien handelt es sich um Zip-Container. Ein solcher Container beinhaltet sowohl mehrere XML-Dateien, die das Dokument vollständig beschreiben, als auch zusätzliche Dateien wie Bilder oder Videos, die in dem Dokument eingebettet sind. Den Zip-Container kann man leicht sichtbar machen, indem man die Dateiendung einfach in .zip ändert.

Der Inhalt eines Dokuments lässt sich nun ändern, indem die entsprechenden XML-Dateien bearbeitet werden. Mithilfe von Open XML SDK ist es möglich die komplexe innere Struktur der XML-Dateien zu überblicken[[4]](#footnote-4).

Hinweis: Das Open XML SDK liegt aktuell in der Version 2.5 vor und kann im Microsoft Download Center kostenlos heruntergeladen werden: <http://www.microsoft.com/en-us/download/details.aspx?id=30425>

Das Open XML SDK stellt zahlreiche Typen und Klassen für die Arbeit mit Dokumenten im Open XMLFormat bereit. Anstatt manuell mit Zip und XML-Dateien arbeiten zu müssen, lassen sich Dokumente mit dem Open XML SDK verhältnismäßig einfach und mit einigen Zeilen Code bearbeiten.

So ist es lassen sich auch In einem Excel-Dokument, das im XML-Format gespeichert wird, XML-Daten als „Custom XML Parts“ unterbringen, die nicht angezeigt werden.

### Custom XML Parts

Über sogenannte benutzerdefinierten XML-Abschnitten (Custom XML Parts) gibt es die Möglichkeit, beliebige Daten in einem Office-Dokument zu speichern, die beim Laden des Dokuments nicht angezeigt werden. Diese Daten werden dann automatisch beim Speichern des Office-Dokuments in den Zip-Container gespeichert und kann später wieder verwendet werden.

# Einführung in Exquisite

## Das Excel-Add-In (Client)

## Der Exquisite-Server

## Die Datenhaltung und Kommunikation

Teil II:  
Exquisite Systemstruktur

# Konzeption und Vorüberlegungen

## Das Client-Server-Modell

## XML als Schnittstelle und neutrales Sicherungsformat

## Choco als Constraint-Solver

# Der Client

Ein Excel-Add-In auf Anwendungsebene bildet den Client des verteilten Programms. In diesem Kapitel werden zunächst die Systemanforderungen und Voraussetzungen beschrieben. Dies soll aufzeigen, welche Systembestandteile zwingend notwendig sind, um eine prinzipielle Lauffähigkeit der Exquisite-Software zu erreichen. Anschließend werden die Werkzeuge, die Installationsschritte und die Konfigurationsmöglichkeiten genauer erläutert. Die Angaben zur empfohlenen Konfiguration beruht auf Erfahrungswerten aus dem Praxiseinsatz.

## Systemanforderungen und Voraussetzungen

Das System wurde in der folgenden Umgebung entwickelt und getestet:

* Windows 7 x64 mit Service Pack 1
* Office (Excel) 2010 x86
* Visual Studio 2010 mit Service Pack 1

Höchstwahrscheinlich wird die Verwendung höherer Versionen von Windows, Visual Studio und Office problemlos möglich sein. Generell lassen sich aber über andere Versionen keine genaue Aussagen treffen, da das Programm auf anderen Systemen nicht getestet wurde.

Der Einsatz einer Java-Laufzeitumgebung ist für den Client nicht zwingend erforderlich. Für die Nutzung des kompletten Funktionsumfangs (Diagnose) wird er aber dringend benötigt. Weitere Informationen zu diesem Thema finden Sie unter anderem im Abschnitt 7.1.

## Werkzeuge

Für die reibungslose Weiterentwicklung des Programms werden folgende Werkzeuge benötigt:

* Visual Studio 2010 Professional oder höher (mit Service Pack 1)
* AnkhSVN[[5]](#footnote-5) – Subversion Unterstützung für Visual Studio
* Office (Excel) 2010
* Optional: R# ReSharper[[6]](#footnote-6) – ReSharper wertet Visual-Studio um diverse Funktionen auf, die vor allem beim Entwickeln sehr hilfreich sind

Wir haben uns für die Verwaltung des Quellcodes in Visual Studio für AnkhSVN entschieden, da es kostenlos ist, vollständig in Visual Studio integriert ist und einige interessante Features bietet.

Hinweis: Die kostenlosen Versionen von Visual Studio (die Express Editionen) bieten keine Unterstützung für VSTO

Hinweis: Es können mit Visual Studio nur Lösungen für Office-Versionen entwickelt werden, die auch lokal auf dem Entwicklungsrechner installiert ist. Eine parallele Installation von verschiedenen Office-Lösungen wird von VSTO nicht unterstützt.

## Die Installation

### Installationsdateien überprüfen

Es empfiehlt sich immer, die Installationsdateien auf Richtigkeit und Vollständigkeit zu überprüfen.

Hinweis: Visual Studio 2010 Ultimate ist auf dem LS13-Server unter „Technical stuff\Software\Visual Studio 2010 Ultimate*“* zu finden (Deutsch und English).

Hinweis: Office Professional Plus 2010 finden Sie auf dem LS13-Server unter *„*Technical stuff\Software\Office Software\Office Professional Plus 2010 update*“*

Hinweis: Die neusten Versionen von AnkhSVN und ReSharper finden Sie unter den jeweiligen Webseiten (siehe Abschnitt 6.2).

### Der Installationsverlauf

Prinzipiell sollten jeweils alle für die verwendeten Windows Version verfügbaren Service Packs installiert sein.

Falls Sie es noch nicht getan haben, melden Sie sich für die Installation bitte als Administrator auf Ihrem System an.

#### Visual Studio 2010 installieren

Ist bereits eine (ältere) Version von Visual Studio installiert, beenden Sie alle zugehörigen Programme. Es empfiehlt sich die bereits installierte Version von Visual Studio vor der Installation von Visual Studio 2010 vollständig zu deinstallieren. Darüber hinaus sollten Sie lieber vorhandene Virenscanner für die Dauer der Installation deaktivieren.

Hinweis: Eine offizielle Anleitung zur Installation von Visual Studio finden Sie hier:  
<http://msdn.microsoft.com/de-de/library/vstudio/e2h7fzkw%28v=vs.100%29.aspx>

#### Office (Excel) 2010 installieren

Ist bereits eine (ältere) Version von Microsoft Office installiert, beenden und deinstallieren Sie alle zugehörigen Programme. Office 2010 lässt sich am leichtesten installieren, wenn es auf dem System vorher noch keine Office-Version installiert war.

Hinweis: Eine Schritt-für-Schritt-Anleitung zur Installation von Office 2010 finden Sie unter: <http://www.office2010-hilfe.de/knowledge/kb_show.php?id=51>

#### AnkhSVN installieren

Beenden Sie Visual Studio und alle zugehörigen Programme vor der Installation von AnkhSVN und starten Sie die Installation mittels Doppelklick auf die AnkhSVN Setup-Datei. Starten Sie anschließend Visual Studio (neu).

Hinweis: Weitere Anleitungen und Informationen zu AnkhSVN finden Sie unter: <http://help.collab.net/index.jsp?topic=/com.collabnet.doc.anksvn_001/action/ankh_getting_started.html>

## Konfiguration

In den folgenden Abschnitten werden die Konfiguration der installierten Anwendungen und Einrichtung des Exquisite-Projektes Schritt für Schritt zum Nachvollziehen beschrieben:

#### SVN-Repository einrichten

Gehen Sie wie folgt vor, um in Visual Studio mittels AnkhSVN ein neues Repository einzurichten:

1. Legen Sie in Visual Studio zuerst ein neues Repository an. Dazu klicken Sie auf Ansicht und dann auf Repository Explorer (siehe Abbildung 6‑1).

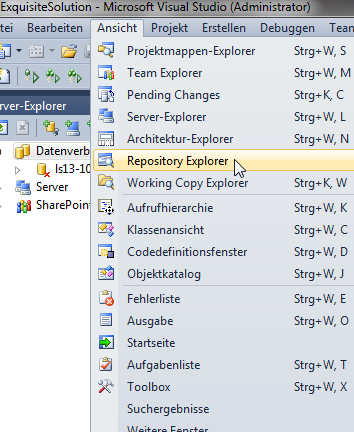


Abbildung 6‑1: Repository Explorer aufrufen

1. Fügen Sie eine neue SVN-URL in den Repository Explorer (siehe Abbildung 6‑2).

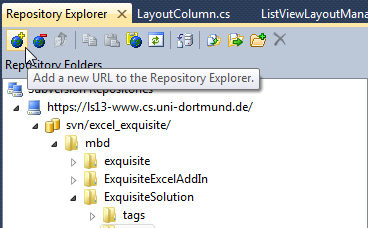


Abbildung 6‑2: Eine neue URL hinzufügen

#### Projekt auschecken

Gehen Sie wie folgt vor, um die Exquisite-Projektmappe aus dem LS13-SVN-Server in Visual Studio einzuchecken.

1. Geben Sie „https://ls13-www.cs.uni-dortmund.de/svn/excel\_exquisite/“ ein und bestätigen Sie Ihre Angaben mit OK.
2. Geben Sie bei Bedarf Ihre SVN-Zugangsdaten ein.
3. Es ist nun möglich das Repository zu durchsuchen. Wählen Sie den Ordner „/excel\_exquisite/mbd/ExquisiteSolution/trunk/“ und öffnen Sie die Projektmappe „ExquisiteSolution.sln“ mit einem Doppelklick (siehe Abbildung 6‑3).

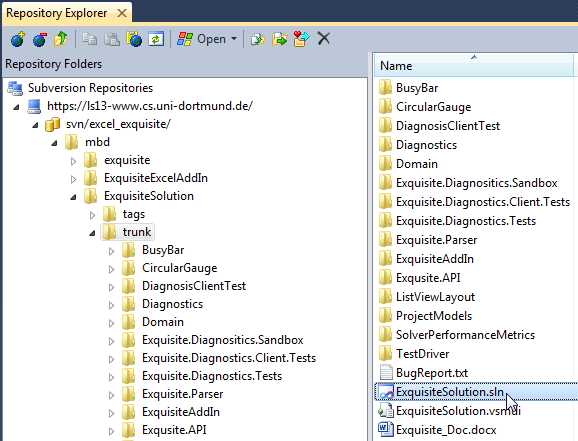


Abbildung 6‑3: Projektmappe auswählen

1. Die Exquisite Projektmappe und alle dazugehörige Projekte sind nun in Visual Studio verfügbar und können bearbeitet und ausgeführt werden.

Hinweis: Die SVN-Befehle befinden sich im Kontextmenü zum Projekt oder den Projektdateien. Wichtige Befehle:

* Commit: Hiermit werden aktuelle Veränderungen in das Repository hochgeladen.
* Revert: Hiermit werden aktuelle Veränderungen in Visual Studio zurückgesetzt.
* Update: Hiermit werden die Dateien auf die aktuelle Version (Head) oder eine ausgewählte Version synchronisiert.

Hinweis: Die Schaltflächen „Revert“ und „Commit“ erscheinen nur bei Veränderungen.

#### Startprojekt festlegen

Um das Excel-Add-In aus Visual Studio heraus starten zu können, muss das entsprechende Projekt als Startprojekt festgelegt sein. Wählen Sie dazu das Projekt „Exquisite.ExcelAddIn“ mit einem Rechtklick aus und klicken sie den Befehl „Als Startprojekt festlegen“ (siehe Abbildung 6‑4).

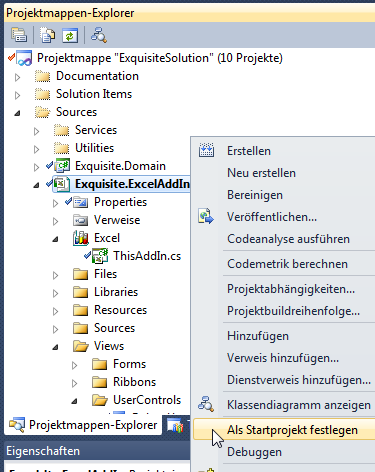


Abbildung 6‑4: Startprojekt festlegen

#### Projekt ausführen

Wenn das Projekt nun aus Visual Studio heraus gestartet wird, wird die dazugehörige Office-Anwendung (Excel) geöffnet und das Add-In in der Anwendung geladen. Beim Beenden der Excel-Anwendung befindet man sich als Entwickler wieder in der Oberfläche von Visual Studio.

Hinweis: Das Add-In ist weiterhin in Excel verfügbar und wird automatisch beim Start von Excel geladen. Ein Add-In auf Anwendungsebene muss also manuell aus der Office-Anwendung entfernt werden, sofern es nicht weiter benötigt wird.

Wenn das Add-In aus Visual Studio heraus gestartet ist, wird mit Hilfe des Kommandos „Clean project“ das Add-In in Excel deaktiviert, sodass der normale Programmstart wie üblich schnell erfolgen kann.

Mehr zum Add-In finden Sie im Abschnitt 8.

# Der Server

## Systemanforderungen und Voraussetzungen

Java JDK 1.7.xx [[7]](#footnote-7)

## Werkzeuge

Eclipse IDE for Java Developers [[8]](#footnote-8)(Juno 64bit has been used so far)

Subversive SVN client plugin for Eclipse[[9]](#footnote-9)

ANT[[10]](#footnote-10)

## Installation

### Der Installationsverlauf

* Ensure the 64bit Java JDK is installed.
* Download ANT and follow instructions from the ANT documentation: <http://ant.apache.org/manual/index.html>

Hinweis: Ensure that there is an environment variable set called ANT\_HOME that points to the location ANT.

* Download the Eclipse IDE and unzip content to e.g. C:\eclipse

Hinweis: Eclipse should be able to run without further modification by clicking on eclipse.exe

Hinweis: The subversive plugin can be installed once Eclipse has been installed. We will discuss this in the next section.

## Konfiguration

Verify Java is installed: Check that C:\Program Files\Java\jdk1.7.xx exists. Then open a cmd window and type java –version. You should then see some text detailing your java install.

Verify ANT is installed: Open a cmd window and type ant –v then hit Enter. If ANT has been installed and configured correctly you should see some text detailing which version of Ant is on your system.

Open Eclipse: since this is the first time it may have been opened, it will prompt you for a workspace folder location. Choose a path to where you want your Eclipse projects to reside.

Install the Subversive SVN plugin: The instructions for installing the plugin are available at:

<http://www.eclipse.org/subversive/installation-instructions.php>

## Build

At this point we should have the various prerequisites installed and working. The next step is to compile and run the exquisite-service project.

Check out the project from SVN. The project is available at: <https://ls13-www.cs.tu-dortmund.de/svn/exquisite-service> you should see a project folder called exquisite-service in the package explorer window of Eclipse.

Hinweis: This URL requires a username and password, contact the system administrator if your access credentials are not working.

The exquisite-service project is used in various ways; firstly as the diagnosis service for the Exquisite Excel AddIn, for investigating various algorithmic techniques for fault diagnosis and generating data to analyze the different approaches under investigation. Therefore there are a couple of different approaches to building and running various parts of project.

This section will focus on what is required to start the diagnosis service so that it is ready to receive requests from clients, i.e. the Excel add-in.

### Build in Eclipse

The server can be run in two ways. The simplest way is to open Eclipse, navigate to the package org.exquisite.communication, and select the class Server, then click run. This should start an instance of Server with any output visible in the output console of Eclipse.

### Build with ANT

The second approach to start the server is to use ANT. To do this, open a cmd window and navigate to the workspace folder where the exquisite-service project resides.

Then cd into the directory called ant.

Run the command ant –f build.xml followed by ant run –f build.xml

With these two commands completed you should see in your cmd window that the server is up and ready to receive requests from clients. Simply press Strg-c to stop the server before closing the cmd window.

This second approach is very useful for invoking the server directly from the Excel client. Additional settings may need to be configured in order for the Excel AddIn to invoke the exquisite-service. These settings are detailed in appendix A.

Teil III:  
Exquisite als integriertes Werkzeug zur Fehlersuche

# Der erste Programmstart

## Der Debug-Mode

* Open the Excel spreadsheet you want to debug. Click on the Exquisite ribbon tab. You should see a set of controls like in Abbildung 6.

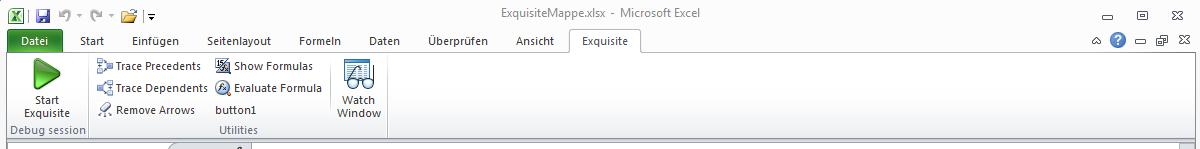


Abbildung 6

* Click on the button labelled Start Exquisite. The Exquisite debug perspective should then be visible, as illustrated in Abbildung 7.

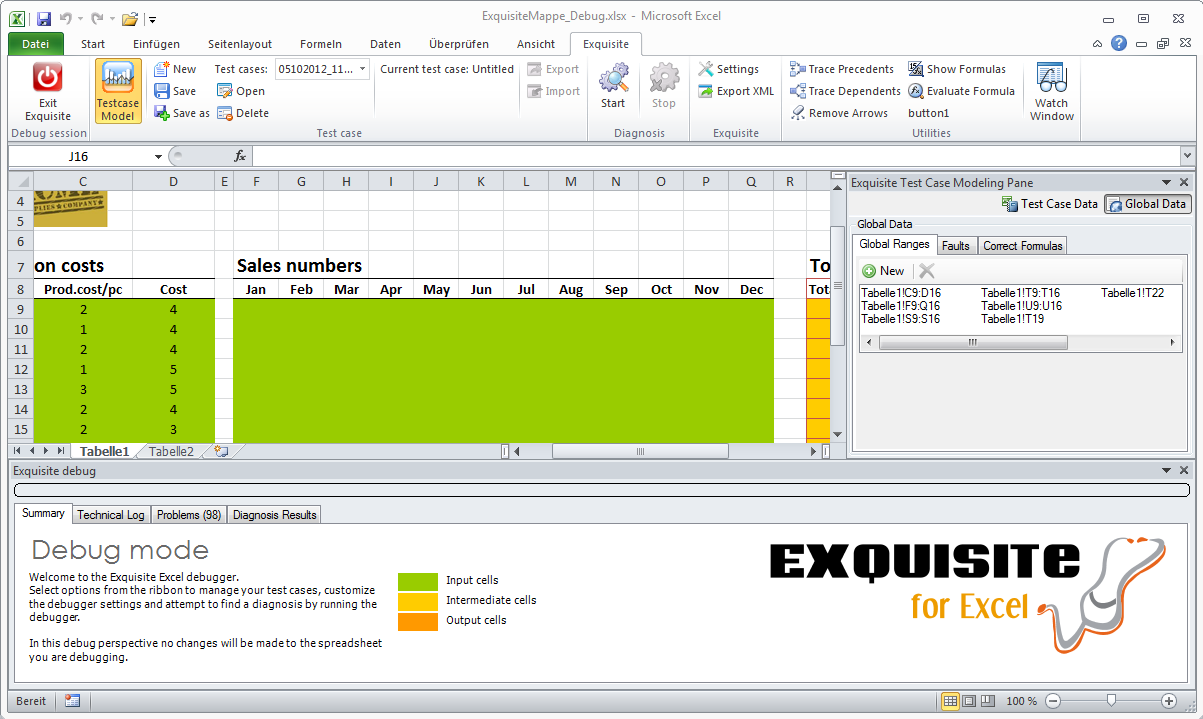
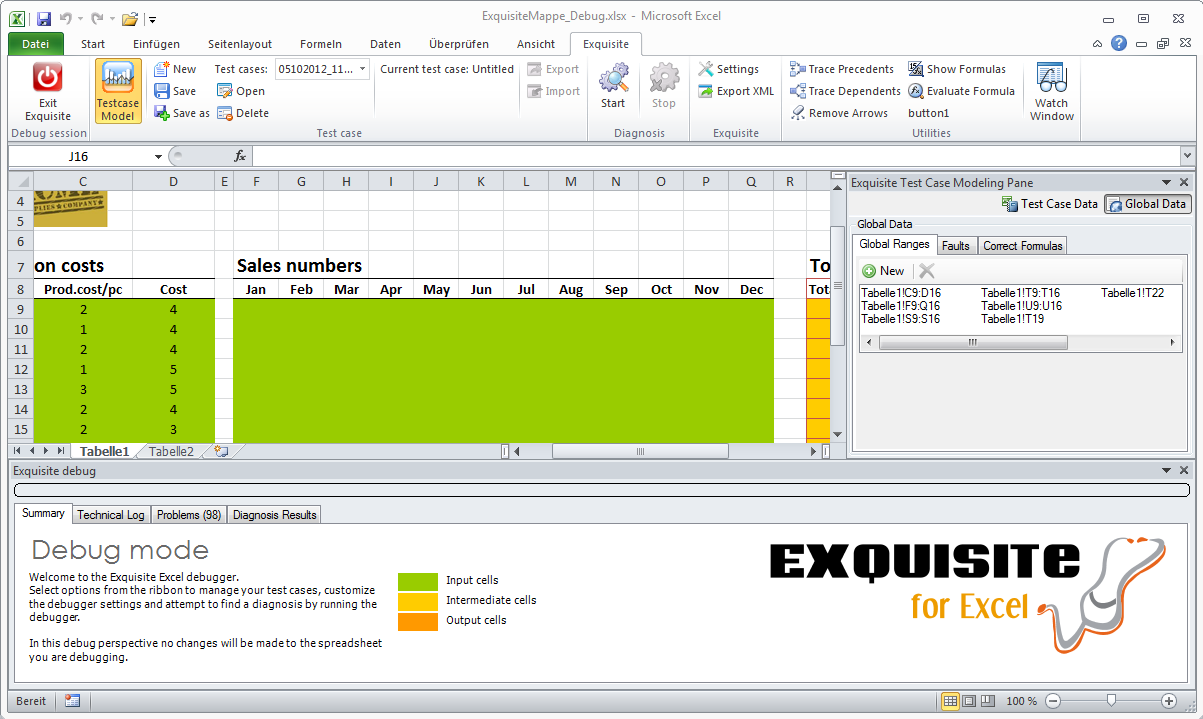


Abbildung 7

# Die Benutzerschnittstelle

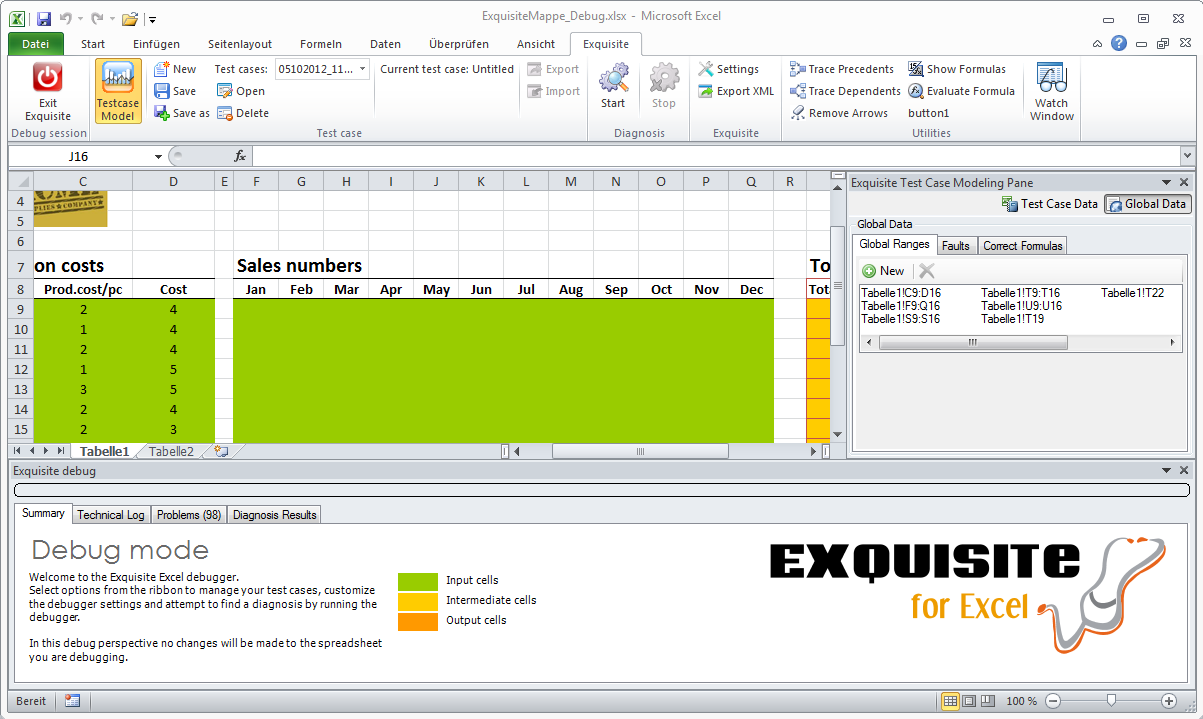
## Exquisite-Menüband (Ribbon)



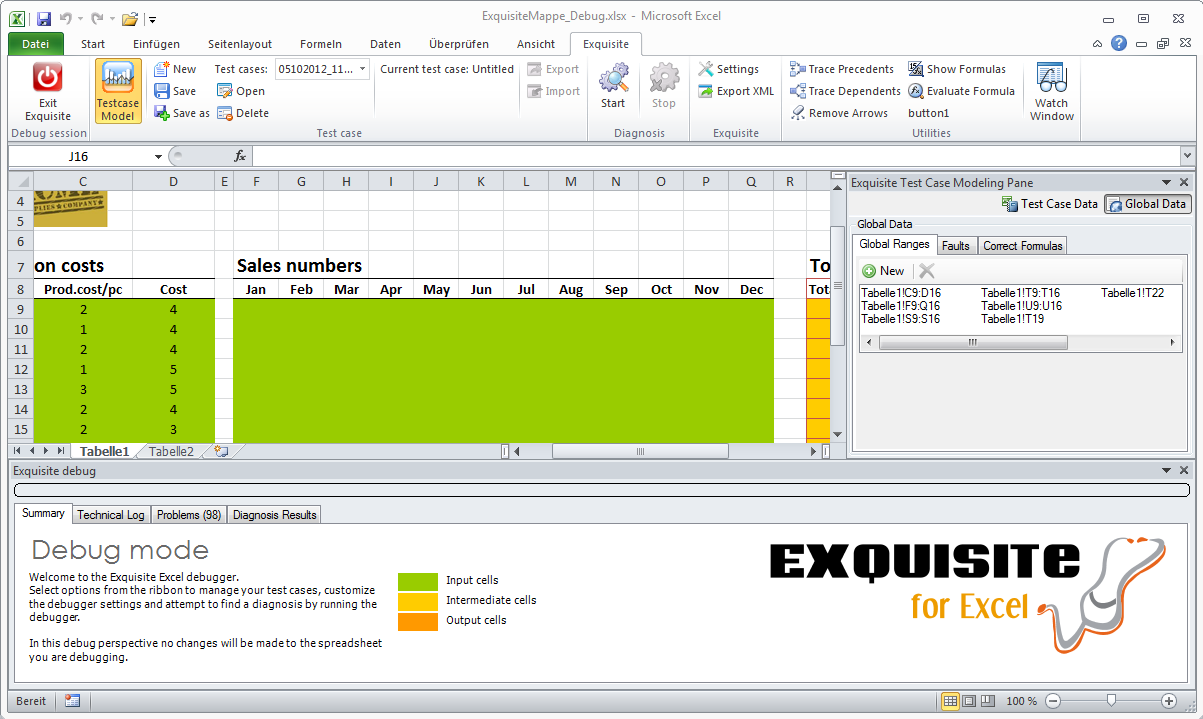
### Testfälle

The Test Case section of the ribbon allows the user to create, update and delete test cases for a given spreadsheet under diagnosis.

### Diagnose

The Diagnosis section contains two buttons to either start the solver to check for faults or to stop an already running solver process.

### Konfigurationsmöglichkeiten

Various configuration options are accessed by clicking on the button. Parameters that can be changed include:

* UI features, such as background colors used by Exquisite to visualize different cell types.
* Default minimum and maximum value bounds of cells.
* Diagnosis parameters, such as the maximum number of diagnoses to return or the maximum search depth the solver is allowed to traverse.
* Localization settings
* Server settings, which are more for development purposes. Enables debug output from the server to be displayed in the server instance console window.

### Werkzeuge (Utilities)

Abbildung 8  
Utilities pane.

A suite of additional tools, some of which are extended versions of existing Excel tools, is also available for the user when diagnosing a faulty spreadsheet.

Trace Precendents: Enables a user to view all the cells that are directly referenced in a formula.

Trace Dependents: Displays all the cells that make reference to a particular cell.

Remove arrow: Clears the spreadsheet of graphical artifacts generated by previous use of Trace Precidents/Dependents.

Show Formulas: Displays the content of all the cells in the spreadsheet that contain a formula.

Evaluate Formula: Enables isolated testing of a selected formula.

## Exquisite-Steuerelemente (UserControls)

### Testfallmodellierung

Abbildung 9  
Test case modeling pane.

The test case modelling pane contains all the controls for creating and editing test cases. Data used for modeling test cases can be scoped in either global or local contexts. Modelling data with a global scope will be persistent for the entirety of the debugging process, regardless of which test case might be in use at a given time. In comparison, data scoped locally is only for the current test case being modeled.

### Debug-Fenster und visuelle Feedbacks

# Diagnose

Some steps to run through to start the diagnosis process.

## Diagnose Starten

After modelling some test cases for the spreadsheet under investigation. Click on the start diagnosis button. The time taken to search for a diagnosis varies depending on the size/composition of the spreadsheet. Therefore you may seen a green activity bar animate across the screen while the diagnosis process is taking place.

## Ergebnisse der Diagnose

Once the diagnosis process has finished, any results returned will be displayed in the diagnosis results tab located at the bottom of the screen.

Teil IV:  
Schlussbemerkungen

# Häufige Fehler

# Häufig gestellte Fragen

Anhang

# Entwicklungsumgebung

#### Starting the Exquisite-Service project from the Excel AddIn project

For various reasons the implementation of the diagnosis solver had to be carried out in Java. This has led to a situation where there are two different development environments involved in the Exquisite project: The Java environment for the solver and the .Net environment for the Excel plugin.

When performing some integration testing or running the system as a whole, it quickly becomes obvious that manually invoking both projects is laborious.

To mitigate this issue, an ANT script was added to the Java project that will automatically compile, JAR and run the server instance. In turn, this ANT script can be called from the Visual Studio client via a batch file that is located in the Files folder of the ExquisiteAddIn project.

This set up enables a programmer working on the ExquisiteAddIn to psuedo-remotely launch the Java solver project.

In order for the ExquisiteAddIn to find the ANT file to compile and run the Java project an additional configuration file must be present on the developers’ machine. We need this configuration file in order to ensure the remote startup of the Java project works on different developers’ machines, which may have project folders and other references structured differently to others.

For convention, each developer working on Exquisite has a folder at the root of C:\ called working. In this folder is contained the config.bat file that will need to be updated to reflect the structural organization of the project files and folders on that particular developers PC. Once this file has been updated the Visual Studio ExquisiteAddIn project will be able to “remotely” invoke the Java diagnosis server.

1. Create a folder (if not already there) C:\working\
2. Open the newly created working folder and create a file called config.bat.
3. Open a text editor and add the following entry to the config.bat file:

echo on

REM needed to explicitly set a path to ant home in order for the .bat file to work in .net

set ANT\_HOME=%ANT\_HOME%

set USER\_HOME=%HOMEDRIVE%%HOMEPATH%

set ECLIPSE\_WORKSPACE=%USER\_HOME%\workspace

set VS2010\_PROJECTS=%USER\_HOME%\Documents\Visual Studio 2010\Projects

#### Die Entwicklungsumgebung besteht aus folgenden Komponenten:

#### Alle lizenzfreien Komponenten befinden sich auf dem LS13-Server.

# Verzeichnisstruktur des Servers

|  |  |
| --- | --- |
| Verzeichnis | Beschreibung |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Verzeichnisstruktur des Clients

|  |  |
| --- | --- |
| Verzeichnis | Beschreibung |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Glossar

|  |  |
| --- | --- |
| Schlüsselwort | Bedeutung |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | Microsoft Corporation, „Office für Entwickler,“ Microsoft Corporation, 03 2013. [Online]. Available: http://msdn.microsoft.com/de-DE/office. [Zugriff am 19 03 2013]. |
| [2] | Microsoft, „Excel Object Model Reference,“ 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff846392.aspx. [Zugriff am 06 03 2013]. |

# Abbildungsverzeichnis

[Abbildung 3‑1: Ausschnitt aus dem Excel-Objektmodell 8](#_Toc350777469)

[Abbildung 6‑1: Repository Explorer aufrufen 18](#_Toc350777470)

[Abbildung 6‑2: Eine neue URL hinzufügen 19](#_Toc350777471)

[Abbildung 6‑3: Projektmappe auswählen 19](#_Toc350777472)

[Abbildung 6‑4: Startprojekt festlegen 20](#_Toc350777473)

# Index

V

VSTO 7

1. Zusätzliche Ressourcen für das Entwickeln von Office-Lösungen: http://msdn.microsoft.com/de-de/library/vstudio/bb772068%28v=vs.100%29.aspx [↑](#footnote-ref-1)
2. Mehr zu PIA: http://msdn.microsoft.com/de-de/library/15s06t57%28v=vs.100%29.aspx [↑](#footnote-ref-2)
3. http://blogs.msdn.com/b/andreww/archive/2007/01/15/vsto-add-ins-comaddins-and-requestcomaddinautomationservice.aspx [↑](#footnote-ref-3)
4. Die offizielle Spezifikation von Office Open XML hat einen Umfang von über 6000 Seiten. [↑](#footnote-ref-4)
5. http://ankhsvn.open.collab.net/ [↑](#footnote-ref-5)
6. http://www.jetbrains.com/resharper/ [↑](#footnote-ref-6)
7. http://www.oracle.com/technetwork/java/javase/downloads/index.html [↑](#footnote-ref-7)
8. http://www.eclipse.org/downloads/ [↑](#footnote-ref-8)
9. http://www.eclipse.org/subversive/ [↑](#footnote-ref-9)
10. http://ant.apache.org/ [↑](#footnote-ref-10)