Installation Secondo for Android

Erstellt im Rahmen der Bachelorarbeit an
der Fernuni Hagen im Wintersemester 2012/2013
von
Jürgen Ehnes

11.08.2013

Version 1.0

Inhalt

1. Übersicht

In diesem Dokument soll die Übersetzung und Installation der Android Version der Secondo Datenbank beschrieben werden. Dazu sind zuerst die Hard- und Softwarevoraussetzungen zu klären. Dazu gehören die Auswahl der Testhardware und auch die Auswahl des Betriebssystems auf dem die Software übersetzt werden soll. Im nächsten Schritt werden die einzelnen Schritte zur Übersetzung der notwendigen Pakete beschrieben und im Anschluss noch die Übersetzung des Programms selbst. Als letztes wird noch beschrieben wie das Programm installiert werden kann.

2. Hardwarevoraussetzungen

2.1. Hardware zum Kompilieren des Programm

Zum Übersetzen des gesamten Programms wird ein PC mit Intel x86 64Bit kompatiblem Prozessor benötigt. Der Prozessor sollte mindestens zwei Cores mit einer möglichst hohen Taktfrequenz haben, da dann die Kompilierung schneller geht. Als Hauptspeicher sollte er mindestens vier, besser sechs Gigabyte RAM haben. Hat der PC weniger Arbeitsspeicher, kann die Kompilierung durch eventuelles Auslagern des Arbeitsspeichers auf die Festplatte übermäßig lange dauern.

Als Massenspeicher sollte eine Festplatte oder noch besser eine SSD verwendet werden, die mindestens 60GB freien Platz hat. Eine SSD führt zu einem schnelleren Kompilerlauf, da die Zugriffszeit deutlich niedriger wie bei herkömmlichen Festplatten ist und somit kleine Dateien schneller gelesen und geschrieben werden können.

Alternativ kann auch eine Virtuelle Maschine genutzt werden. Hierfür bieten sich Produkte wie VMware Workstation oder VirtualBox von Oracle an. Die Nutzung von VMware VSphere oder Microsoft HyperV ist aber ebenso möglich. Bei der Ausstattung der virtuellen Maschine kann man sich an der Ausstattung der realen Hardware orientieren.

2.2. Hardware zum Ausführen des kompilierten Programms

Als Hardware zum Ausführen des kompilierten Programms eignen sich alle Android-kompatiblen Geräte mit ARM v7-kompatibler (oder höher) CPU. Das Gerät sollte mindestens 1GB Arbeitsspeicher und mindestens 1GB freien Flashspeicher besitzen. Eine Dualcore CPU hilft auch hier, aufwendige Berechnungen schneller zu machen.

Beispiele hierfür sind:

- Samsung Galaxy Tab2 (7+10 Zoll) und neuer
- Handys mit Android 4.0 und höher
- Pandaboard (Wenn Android stabil läuft)
- Cubieboard
- Android USB-Sticks mit mind. 1GB RAM

3. Softwarevoraussetzungen

3.1. PC zum kompilieren

Secondo lässt sich auf Linux, MacOSX und Windows kompilieren. Der Android Teil wurde nur auf SUSE Linux 12.2 kompiliert. Ob Windows und MacOSX gehen wurde nicht geprüft. Offiziell gibt es Secondo auch für Ubuntu. Mit kleinen Anpassungen sollte sich die vorliegende Version auch unter diesem Linux kompilieren lassen.

3.2. Zielsoftware

Secondo für Android läuft auf Androidversionen ab 4.0 (ICS) Versionen vor 4.0 werden nicht unterstützt.

4. Installation Suse Linux

Das Betriebssystem sollte auf einer DVD oder einer ISO-Datei vorliegen, je nachdem ob auf einer realen Hardware oder in einer virtuellen Maschine installiert wird.

Zum Installieren wird die DVD ins DVD-Laufwerk eingelegt, bzw. das ISO-File der virtuellen Maschine zugewiesen. Danach den PC starten und im Startmenu den Punkt „Installation“ auswählen. Als erstes erscheint ein Bildschirm auf dem die Sprache und die Tastaturbelegung ausgewählt, sowie die Lizenz abgenickt werden kann.

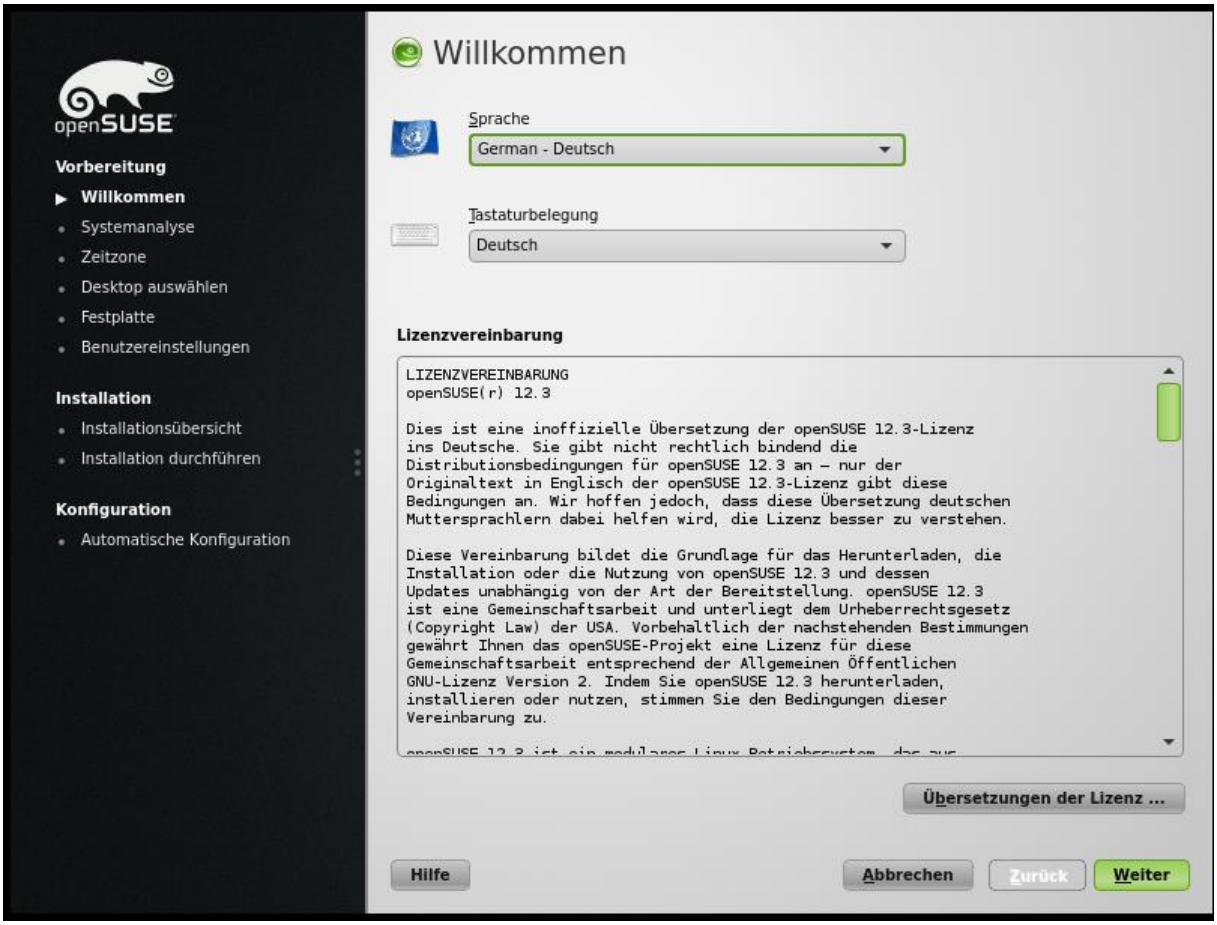

Abbildung 1 - Willkommensbildschirm

Wenn man die richtigen Einstellung gewählt hat und auf „Weiter gedrückt hat, kommt man zum nächsten Bildschirm.

Dort wählt man „Neuinstallation“ wie man in Abbildung 2 sehen kann, aus. Die restlichen Punkte kann man so lassen.

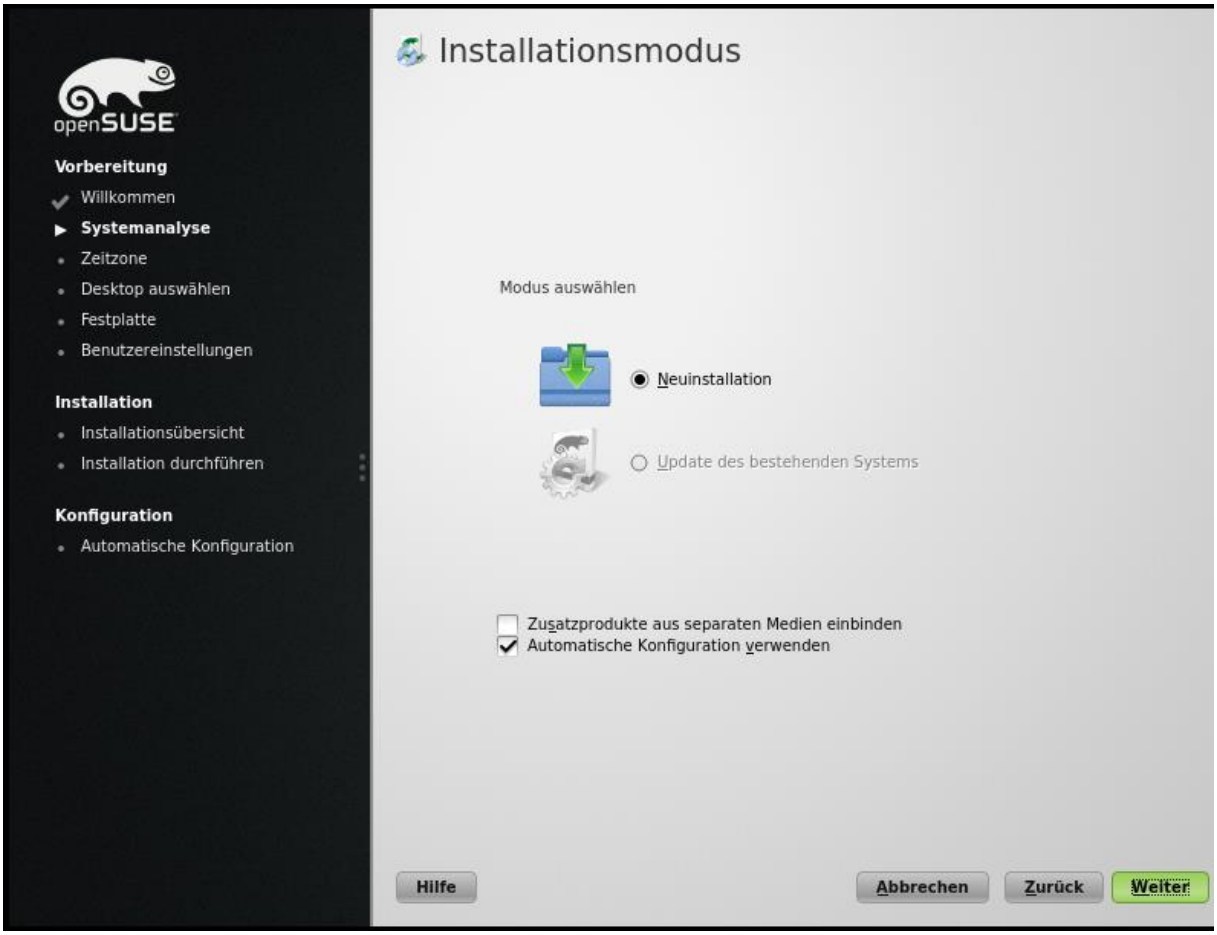

Abbildung 2 - Auswahl des Installationsmodus

Nach dem Drücken von „Weiter“ kommt man zum Bildschirm in Abbildung 3. Hier kann man die Uhrzeit einstellen und die Zeitzone auswählen. Die Einstellung „Rechneruhr auf UTC gestellt“ habe ich hier nicht angehakt, da es bei mir nicht zutrifft. Da eine Änderung an dieser Einstellung aber sofort sichtbar wird, kann man ausprobieren was für den eigenen PC zutrifft. Bei VMs ist die hier gezeigte Einstellung normalerweise richtig.

Den Hinweis in Abbildung 4 kann man im Normalfall ignorieren und mit Fortfahren weiter machen.

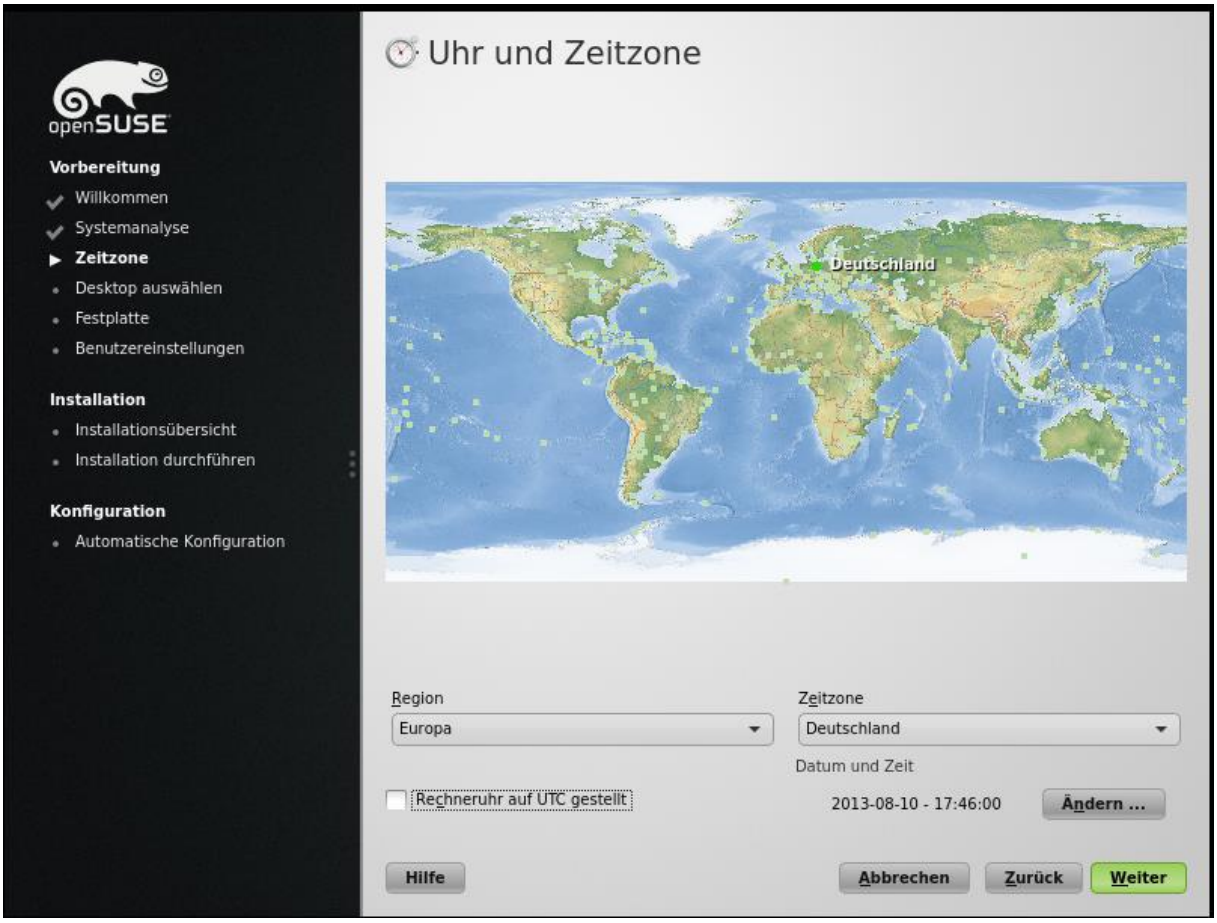

Abbildung 3 - Auswahl Uhr und Zeitzone

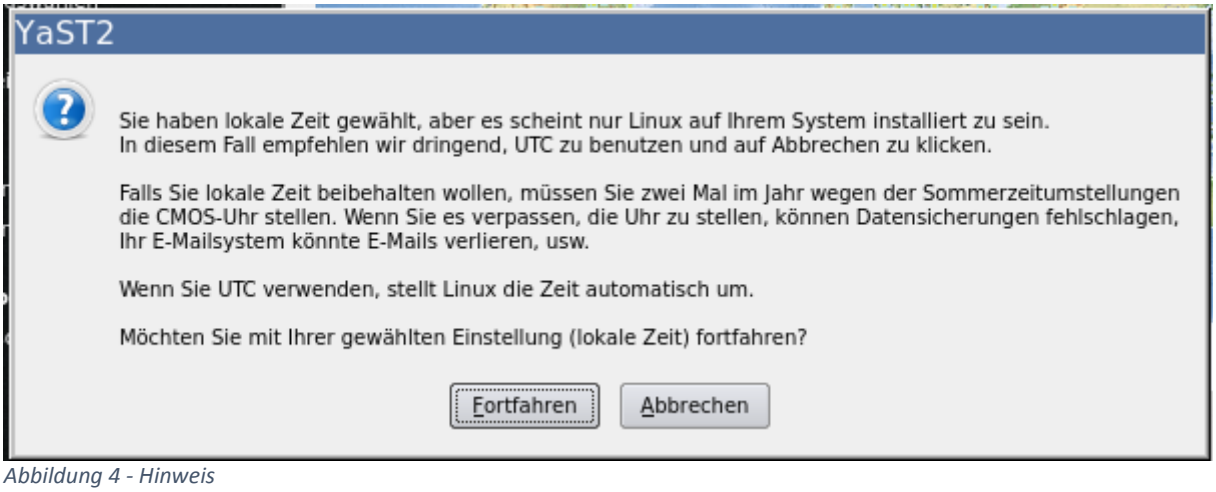

Abbildung 4 - Hinweis

In Abbildung 5 sieht man den Dialog zum Auswählen des bevorzugten Desktops. Dem Autor gefällt die Standardeinstellung KDE-Desktop am besten. Wer mag kann hier auch GNOME wählen.

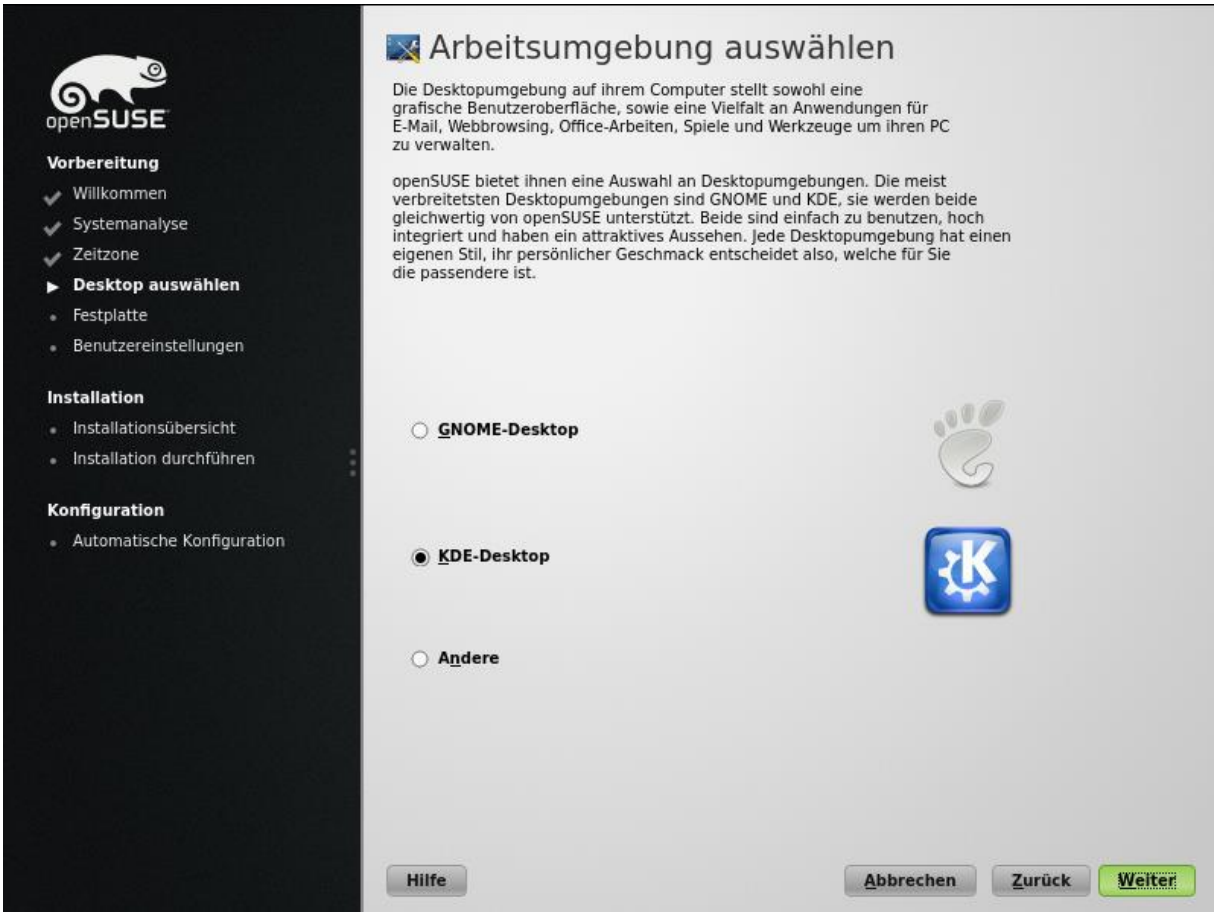

Abbildung 5 - Desktop auswählen

Nach dem Drücken von Weiter kommt man zur Abbildung 6. Hier kann die Partitionierung geändert werden. Bei 60GB Plattenplatz ist die hier vorgeschlagene Partitionierung in Ordnung. Wichtig ist, dass man auf der Home-Partition ausreichend Platz für die Übersetzung von Secondo hat. Falls man mehrere Versionen des Secondo-Verzeichnisses gleichzeitig auf der Festplatte haben möchte, sollte man mindestens 20-30GB Platz haben. Alternativ kann man auch den gesamten Platz der Root-Partition (/) geben in dem man den Haken bei „Separate Home-Partition vorschlagen“ rausmacht.

Für diese Installation übernimmt der Autor einfach den Vorschlag vom Betriebssystem.

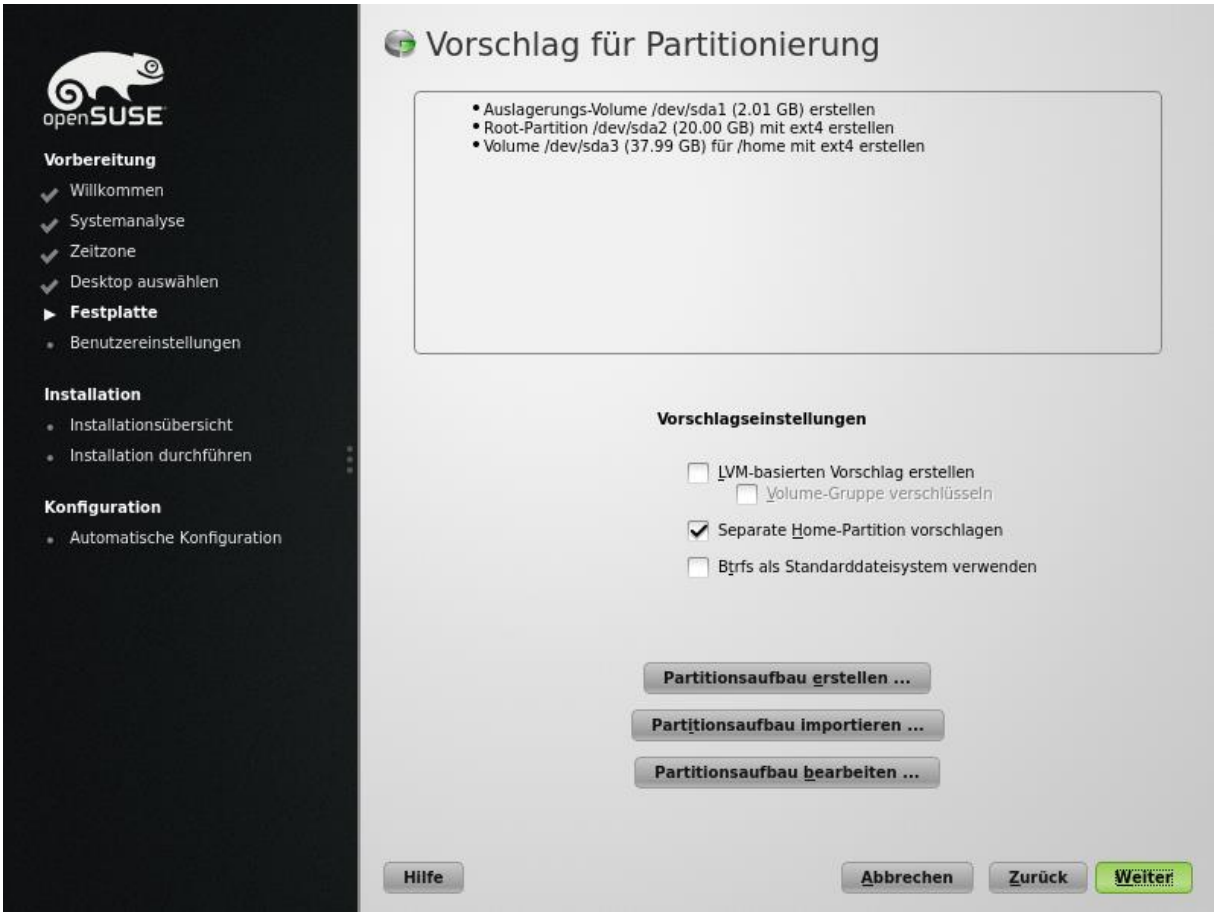

Abbildung 6 - Partitionierung

Nach dem Drücken auf „Weiter“ kommt man zu den Benutzereinstellungen (Abbildung 7).

Hier gibt man einen Benutzernamen und ein Kennwort ein. Man kann wählen, ob man das Passwort auch für den Root-User nutzen will und ob man den erstellten Benutzer automatisch anmelden möchte. In unserem Fall ist das OK.

Wenn der Hinweis aus Abbildung 8 kommt, muss man entscheiden ob man ein anderes Passwort angeben möchte, oder mit dem eingegebenen weitermacht. Wir machen einfach mit „Ja“ weiter.

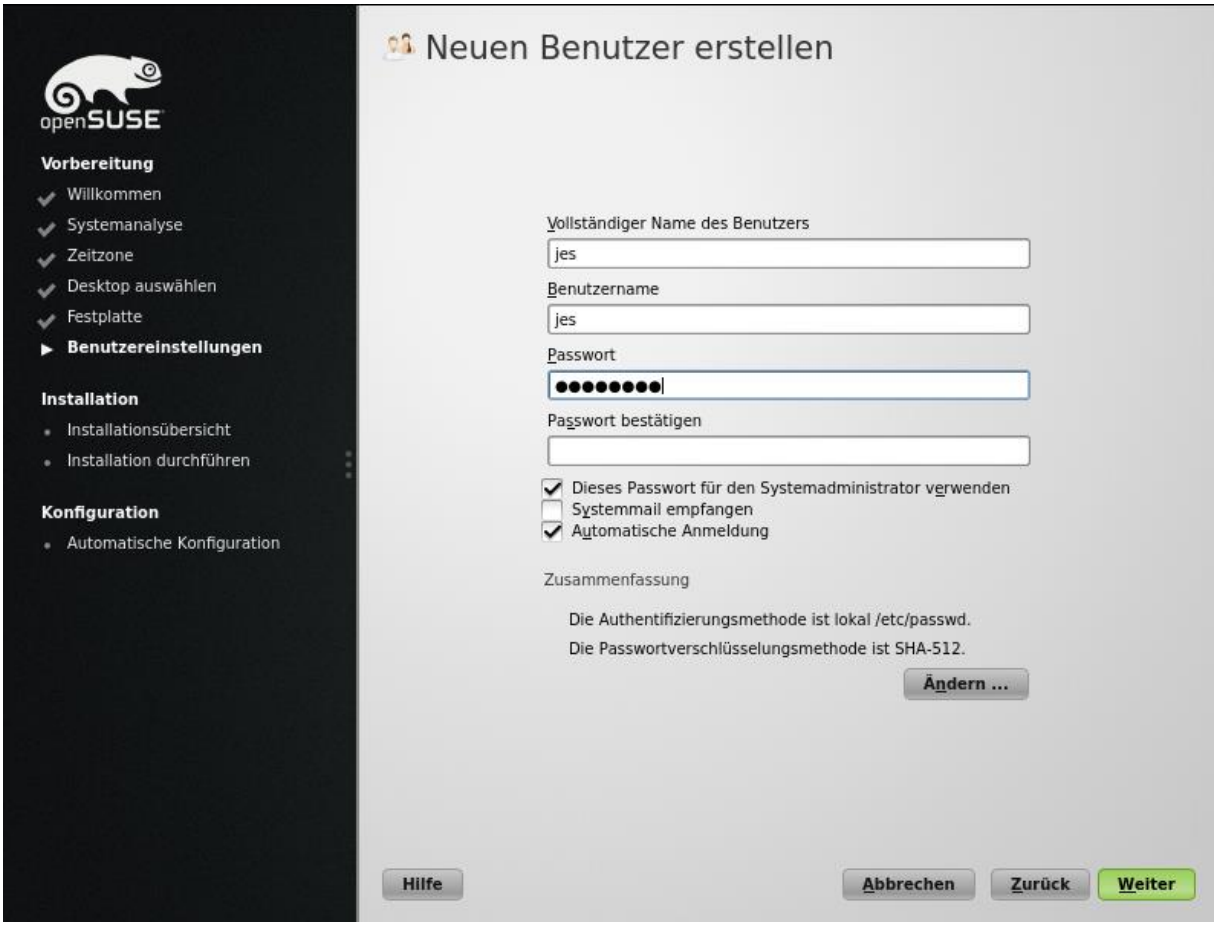

Abbildung 7 - Benutzereinstellungen

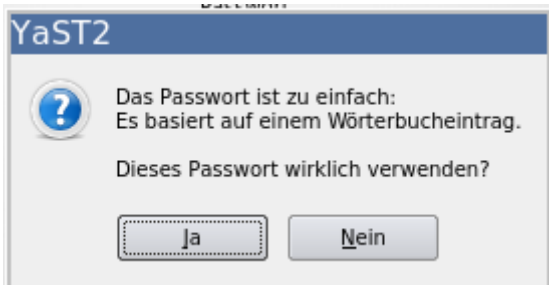

Abbildung 8 - Passwortkontrolle

Nach dem der vorige Dialog erfolgreich abgeschlossen wurde, geht es weiter mit Abbildung 9.

Die Vorauswahl ist für diesen Zweck in Ordnung und es sind normalerweise keine Änderungen nötig. Die zusätzlich benötigten Pakete werden im nächsten Schritt nachinstalliert. Nach Drücken des Buttons „Installieren“ wird man noch einmal aufgefordert zu bestätigen, dass man wirklich installieren möchte. Wenn man dies bestätigt hat, geht es mit der eigentlichen Installation los.

Die Installation dauert nun je nach Geschwindigkeit der Hardware zwischen zehn und 30 Minuten.

Abbildung 9 - Einstellungen für die Installation

Abbildung 10 - Bestätigung

Nach dem die Installation beendet ist, erscheint der Desktop mit einer Hinweismeldung, die man schließen kann(Abbildung 11).

Diesen Desktop kann man sich nun nach Belieben einrichten.

Damit ist die Installation des Betriebssystems beendet.

Abbildung 11 - Desktop

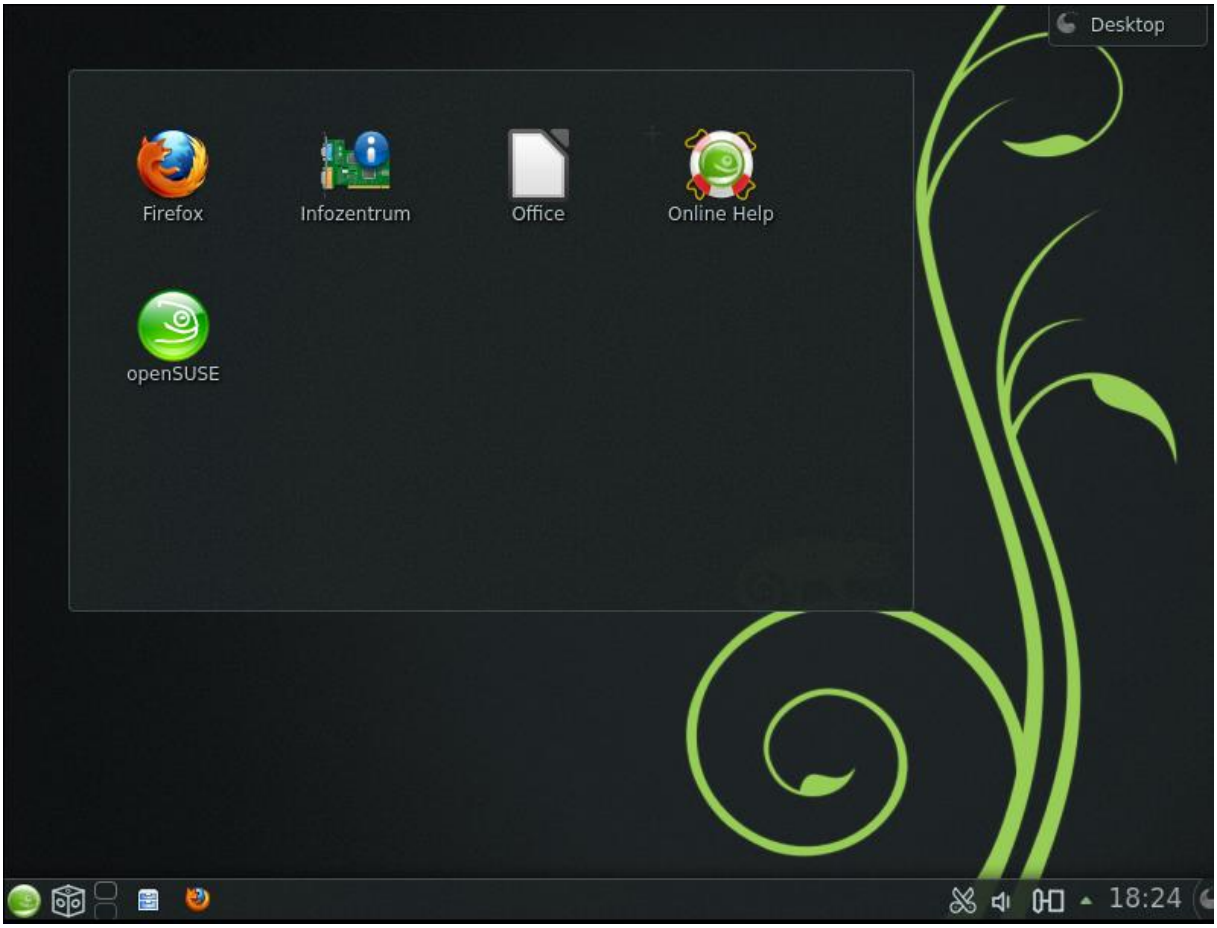

Abbildung 12 - Desktop 2

5. Nachinstallieren der benötigten Pakete

In diesem Schritt müssen weitere benötigte Pakete nachinstalliert werden. Bei OpenSUSE geschieht dies in den Systemeinstellungen.

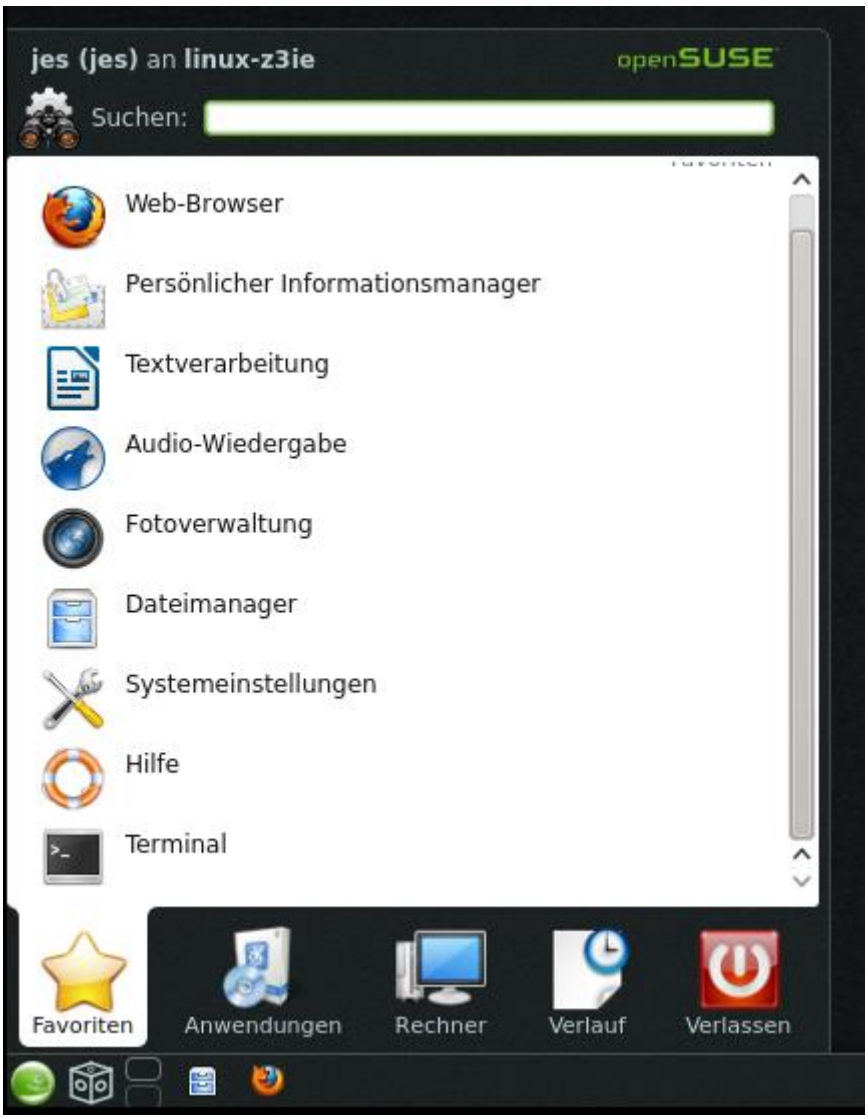

Abbildung 13 - Menu Systemeinstellungen

Dort geht man ganz nach unten und wählt Yast (Abbildung 14)

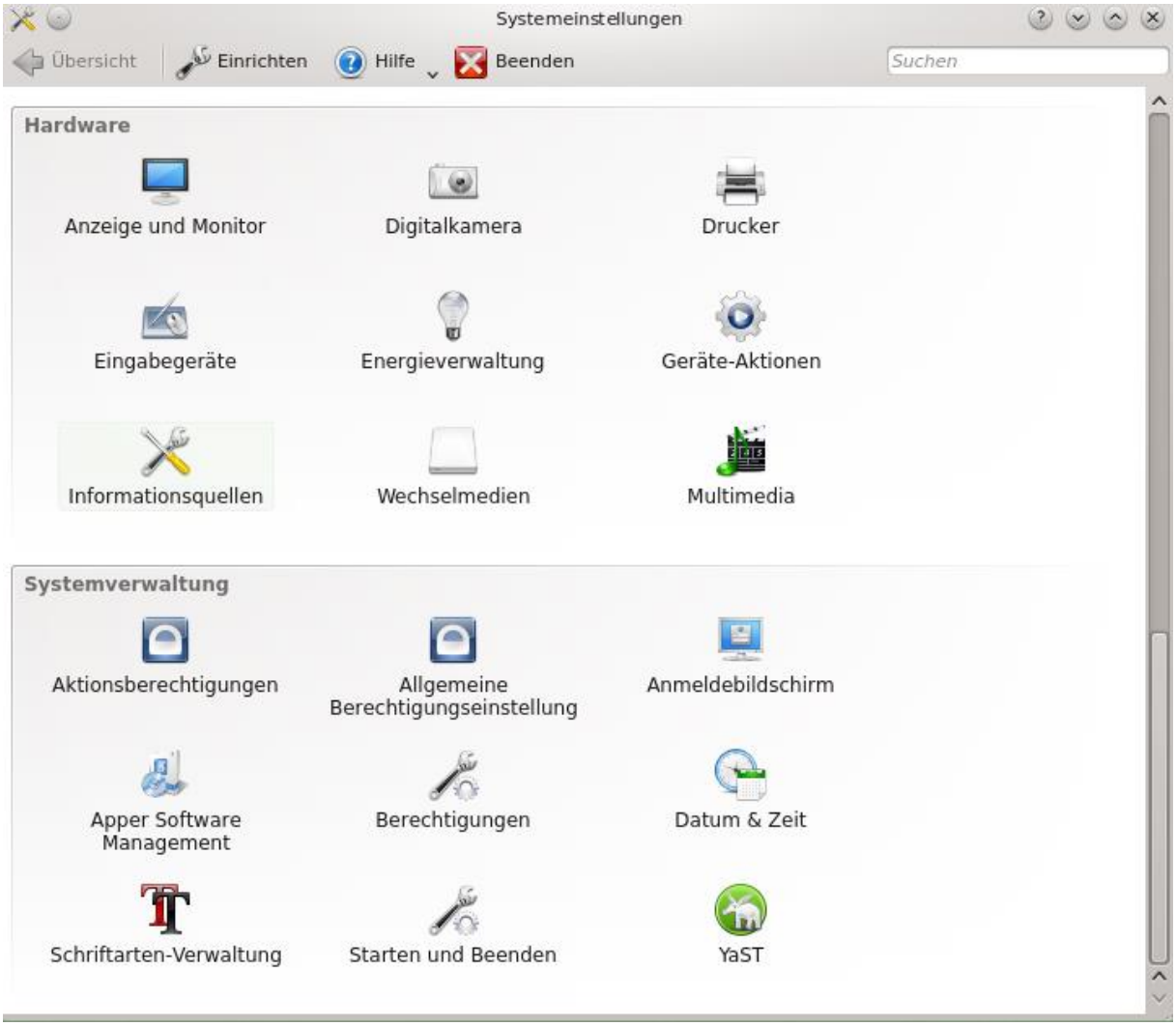

Abbildung 14 - Yast auswählen

Hat man YaST ausgewählt, öffnet sich ein neues Fenster (Abbildung 15). In diesem Fenster muss man das Passwort, das man vorher eingegeben hat nochmal eingeben, damit man System-Administratorrechte erhält.

Abbildung 15 - Passwortabfrage

Nach dem Eingeben des Passworts und Drücken von „OK“. Kommt man zu YaST (Abbildung 16).

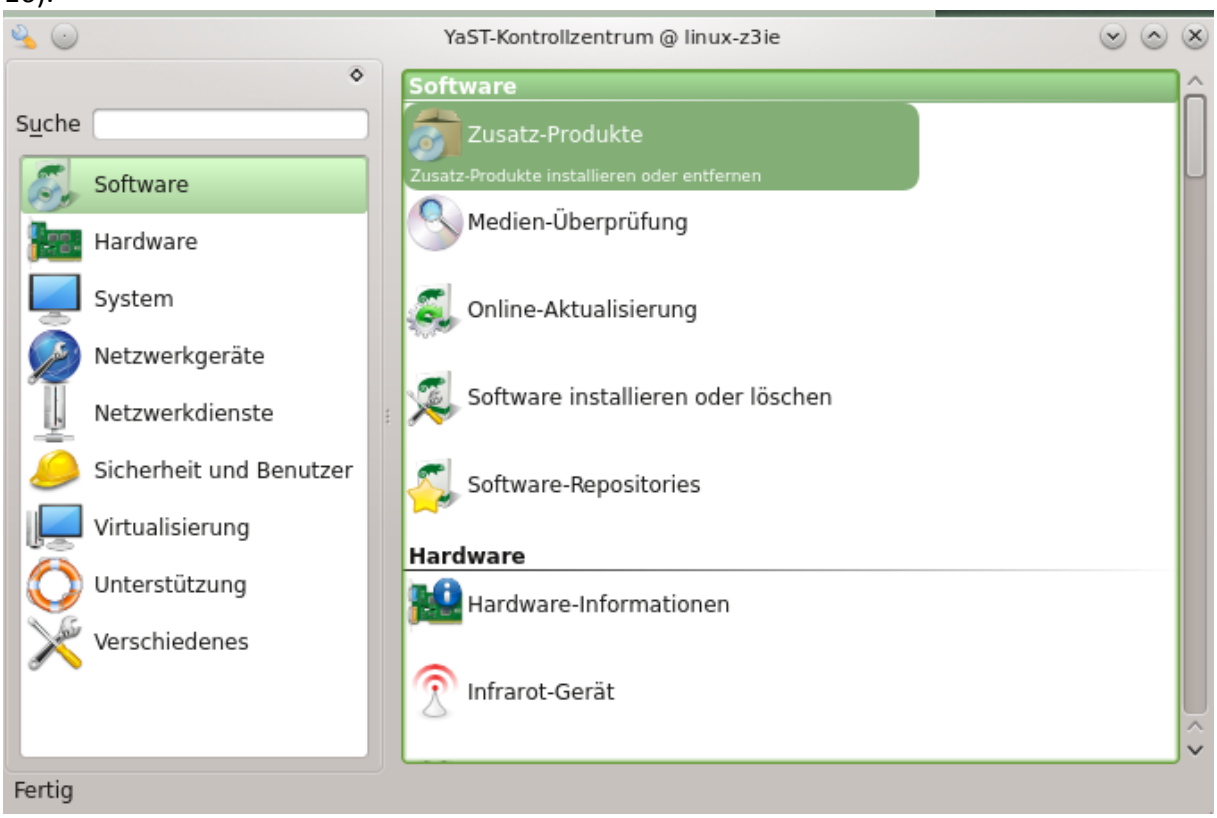

Abbildung 16 - YaST

Hier wählt man in der linken Hälfte „Software“ in der rechten, „Software installieren oder löschen“. Die Warnung aus Abbildung 17 kann man ignorieren.

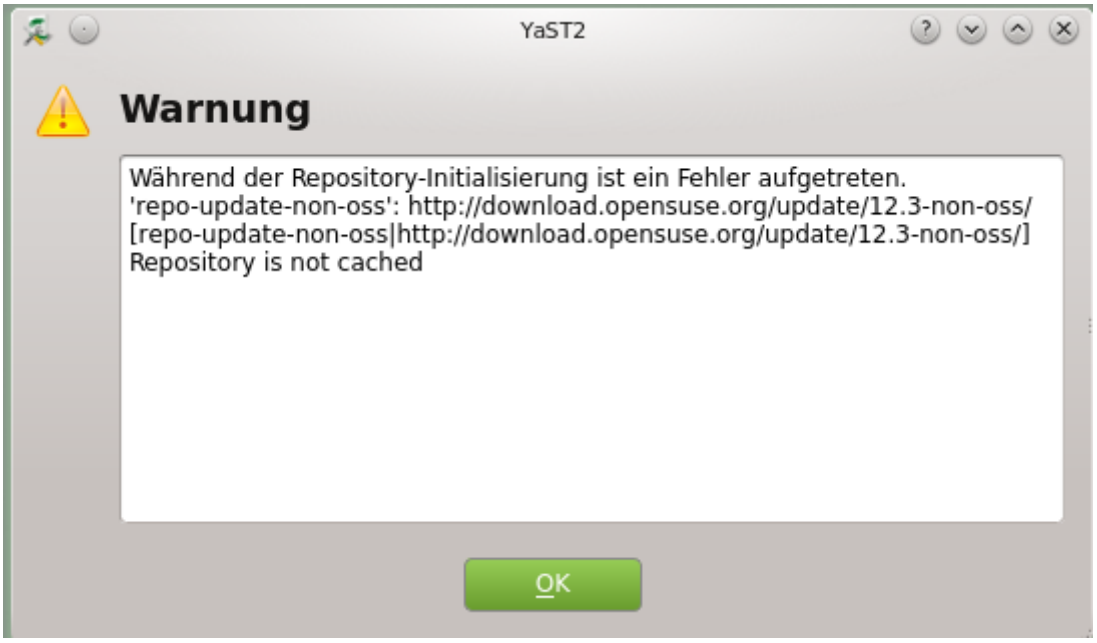

Abbildung 17 - Warnung

In Abbildung 18 sieht man das Programm zu Softwareinstallation. Hier muss man die folgenden Pakete nachinstallieren:

- m4
- make und automake
- bison
- flex
- mc
- gcc
- gcc-c++
- swipl
- java SDK 1.7.0 Devel
- ant

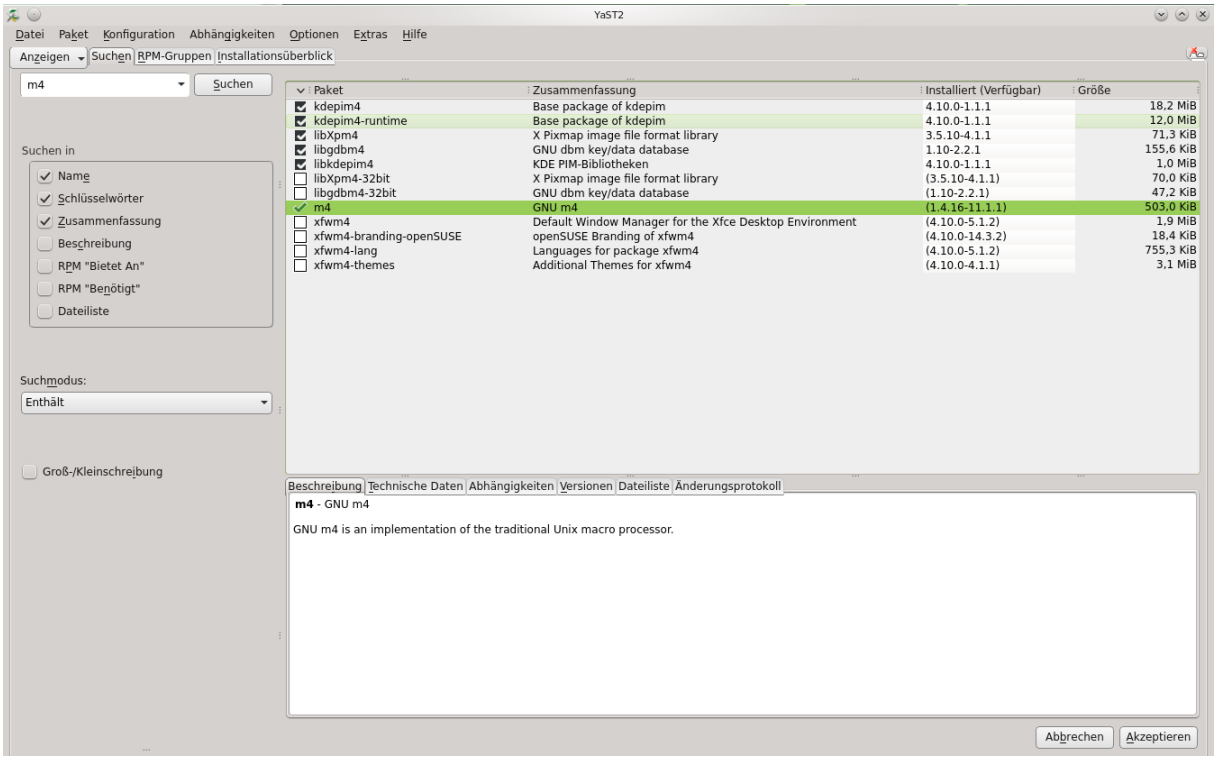

Abbildung 18 - Software auswählen

Nachdem alle nötigen Pakete ausgewählt wurden, kann man mit „Akzeptieren“ die Installation anstoßen. Manchmal müssen zu den ausgewählten Paketen weitere installiert werden. Dies sieht man im Dialog „Automatische Änderungen“ in Abbildung 19. Dies bestätigen und warten bis die Installation beendet ist.

Abbildung 19 - Automatische Änderungen

Nun sollte noch die Online-Aktualisierung ausgeführt werden um die aktuellsten Pakete zu erhalten. Sollte es dabei Probleme geben, bitte überprüfen ob die Netzwerkkarte richtig funktioniert und eine Verbindung zum Internet besteht. Im Zweifelsfall das System durchbooten.

6. Herunterladen der zusätzlichen Pakete

Einige Pakete müssen als ARM-Version kompiliert werden, auch wenn Sie schon als Intel-Version heruntergeladen wurden. Dazu werden diese Pakete aus dem Internet heruntergeladen und im Homeverzeichnis des Users entpackt.

Folgende Pakete müssen separat heruntergeladen werden:
flex 2.5.37: http://flex.sourceforge.net/
bison 2.7: http://www.gnu.org/software/bison/bison.html
Bitte hier unbedingt die Version 2.7 verwenden, die Version 3.0 lässt sich nicht mit der aktuellen Android Entwicklungsumgebung kompilieren

libgsl 1.15: ftp.gnu.org/gnu/gsl/gsl-1.15.tar.gz
libjpeg 6b: http://sourceforge.net/projects/libjpeg/files/libjpeg/6b/
readline 6.2: http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html
berkeley db: http://download.oracle.com/berkeley-db/db-5.3.21.NC.tar.gz
Android SDK 20130219 oder neuer: http://developer.android.com/sdk/index.html
Android NDK 8d oder r9: http://developer.android.com/tools/sdk/ndk/index.html
Eclipse 4.2: http://www.eclipse.org
Eclipse wird nur benötigt, wenn das ADT von Google nicht verwendet wird.

7. Erzeugen der ARM-Toolchain

In diesem Kapitel setze ich voraus, dass alle Downloads im Verzeichnis ~/Downloads liegen. Mit cd ~ begeben wir uns ins Homeverzeichnis des angemeldeten Benutzers.

unzip -x Downloads/adt-bundle-linux-x86_64-20130219.zip
Nach dem Auspacken gibt es ein neues Verzeichnis adt-bundle-linux-x86_64-20130219
Nun erstellen wir einen Link
ln –s ~/ adt-bundle-linux-x86_64-20130219/sdk android-sdk
Um die Bedienung einfacher zu machen gehen wir z.B. mit Dolphin in das Unterverzeichnis adt-bundle-linux-x86_64-20130219/eclipse und ziehen uns eine Verknüpfung auf den Desktop
Dann starten wir Eclipse durch einmaliges Klicken auf das Eclipse Icon

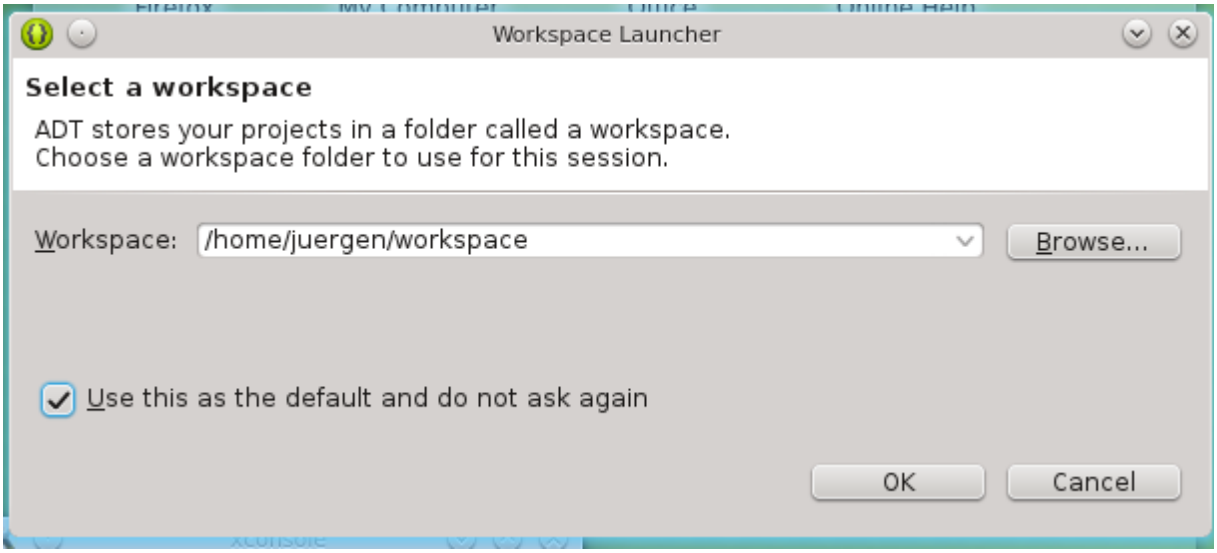

Abbildung 20 - Workspace auswählen

Den Pfad für den Workspace setzen wir beliebig. Und den Haken bei „Use this as the default and do not ask again"
Die restlichen Fragen bitte entsprechend der eigenen Wünsche beantworten.
Nun ist Eclipse erst mal soweit vorbereitet.
Als nächstes wird das SDK fertig eingerichtet

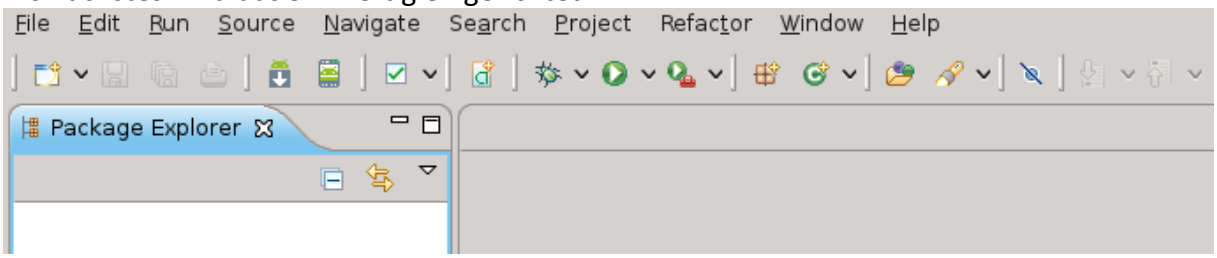

Abbildung 21 - SDK-Manager starten

Dazu wird der Android SDK-Manager gestartet (Abbildung 21)

Dann müssen weitere Pakete für das SDK installiert werden. Das ist in Abbildung 22 zu sehen.

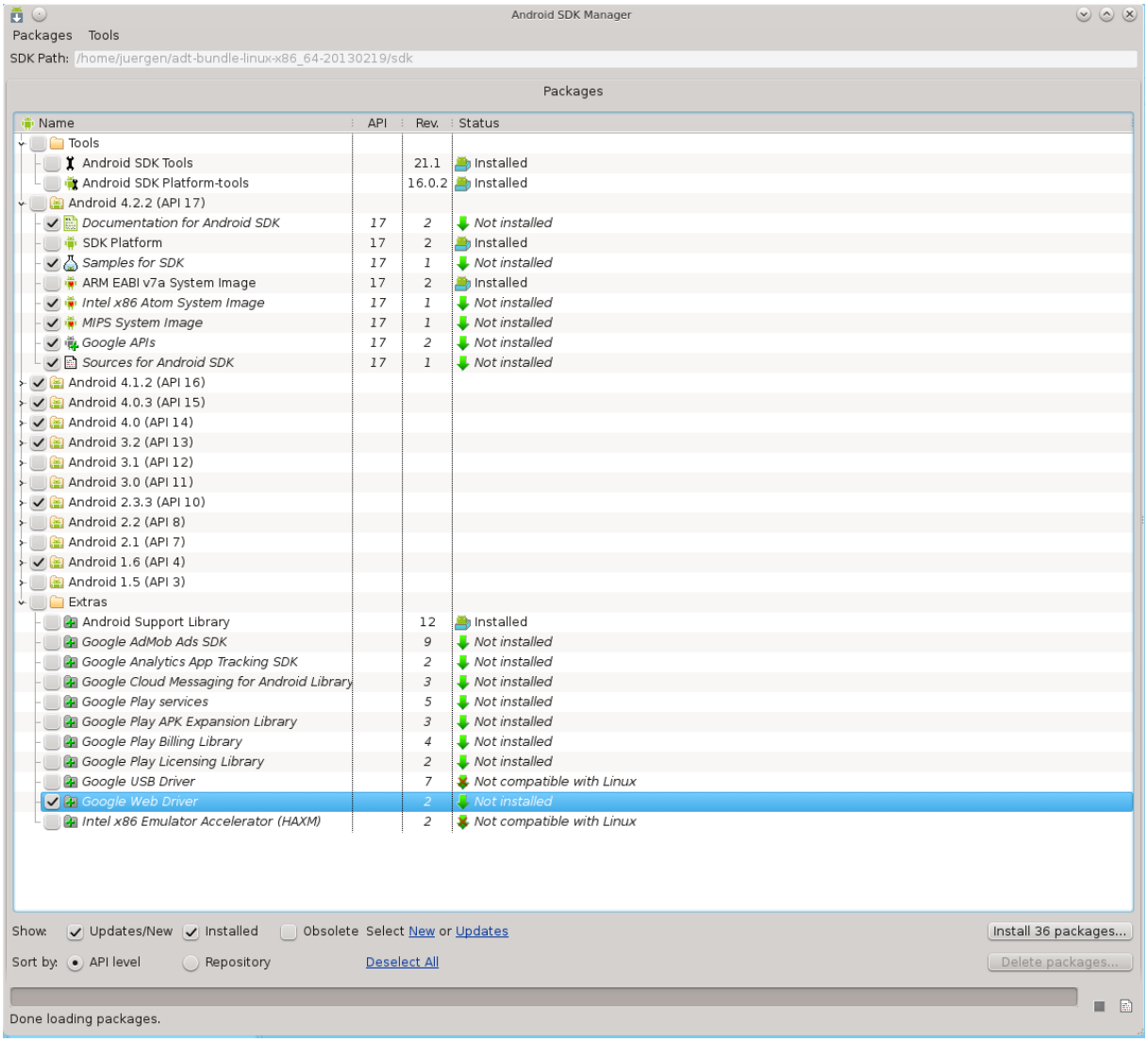

Abbildung 22 - Nachinstallieren der Pakete

An dieser Stelle wird entschieden für welche Plattformen und Android Versionen entwickelt werden können.
Wir benötigen mindestens API 14-17, die SDK-Tools und die SDK-Plattform-Tools. Falls man weiter Versionen von Android unterstützen möchte, kann man hier noch zusätzliche Plattformen auswählen.

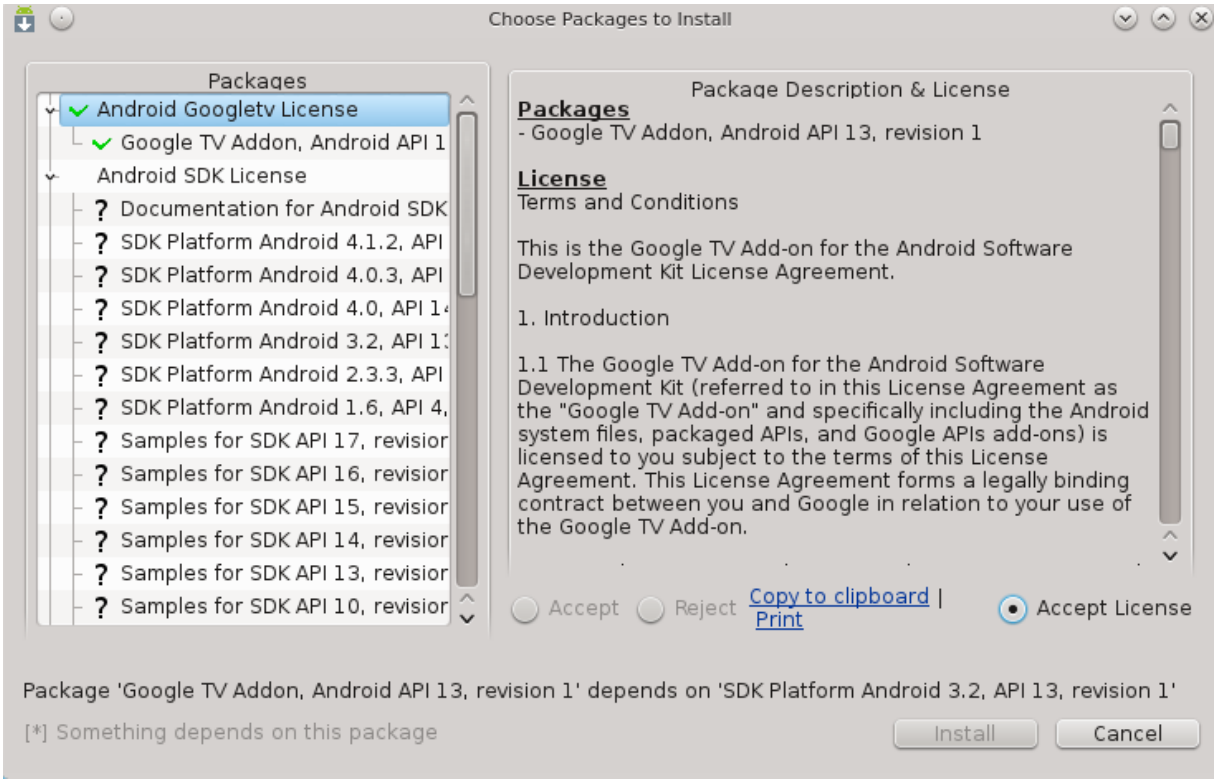

Abbildung 23 - Bestätigen der Lizenz

Nach dem Akzeptieren der Lizenz werden die notwendigen Pakete installiert(Abbildung 23).

Wenn dieser Teil beendet ist, muss ein virtuelles Gerät, ein sogenanntes Android Virtual Device eingerichtet werden
Dieses AVD ist ein Android Emulator, für den Fall, dass zum Testen keine Hardware vorhanden ist.

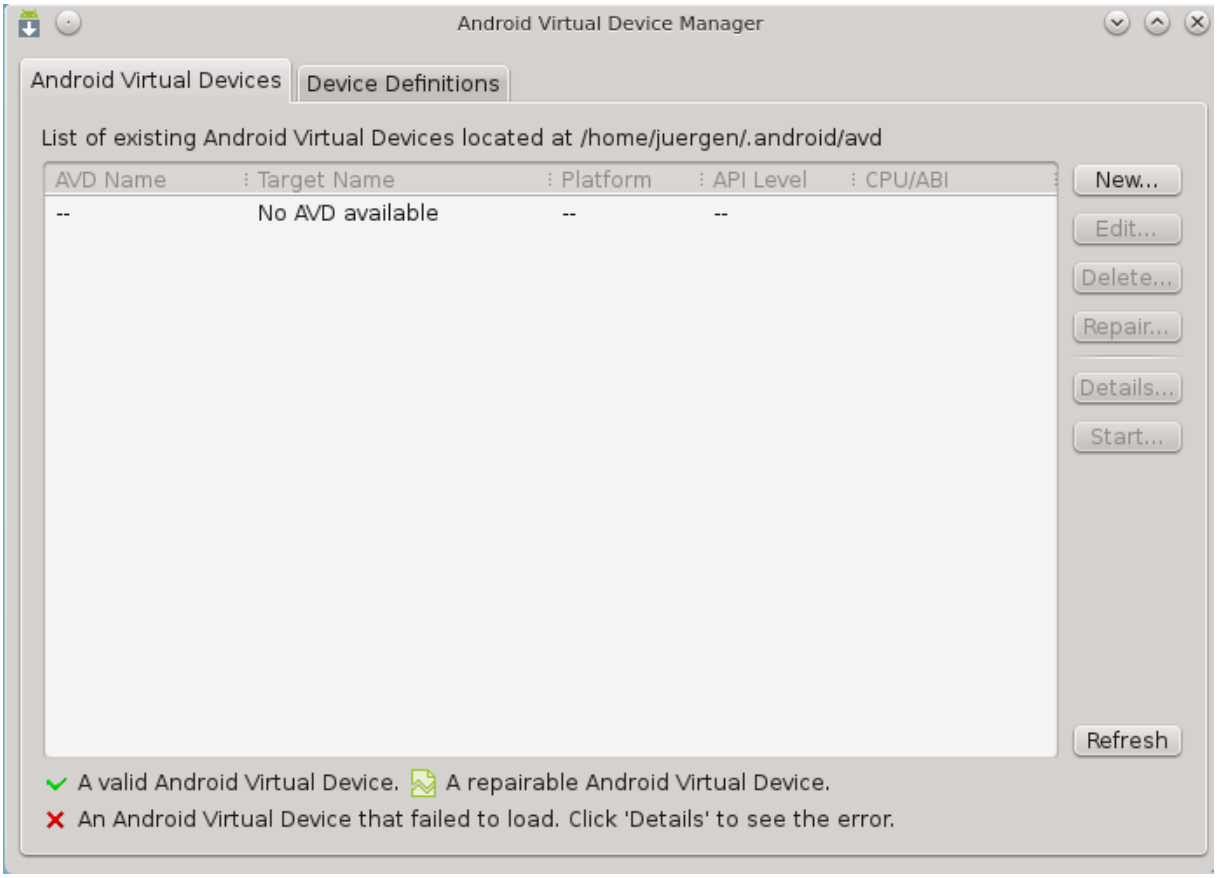

Abbildung 24 - AVD

Dazu öffnen wir den Android Virtual Device Manager(Abbildung 24). Klicken auf New und erstellen eine Maschine, die in etwa so aussieht wie in Abbildung 25.

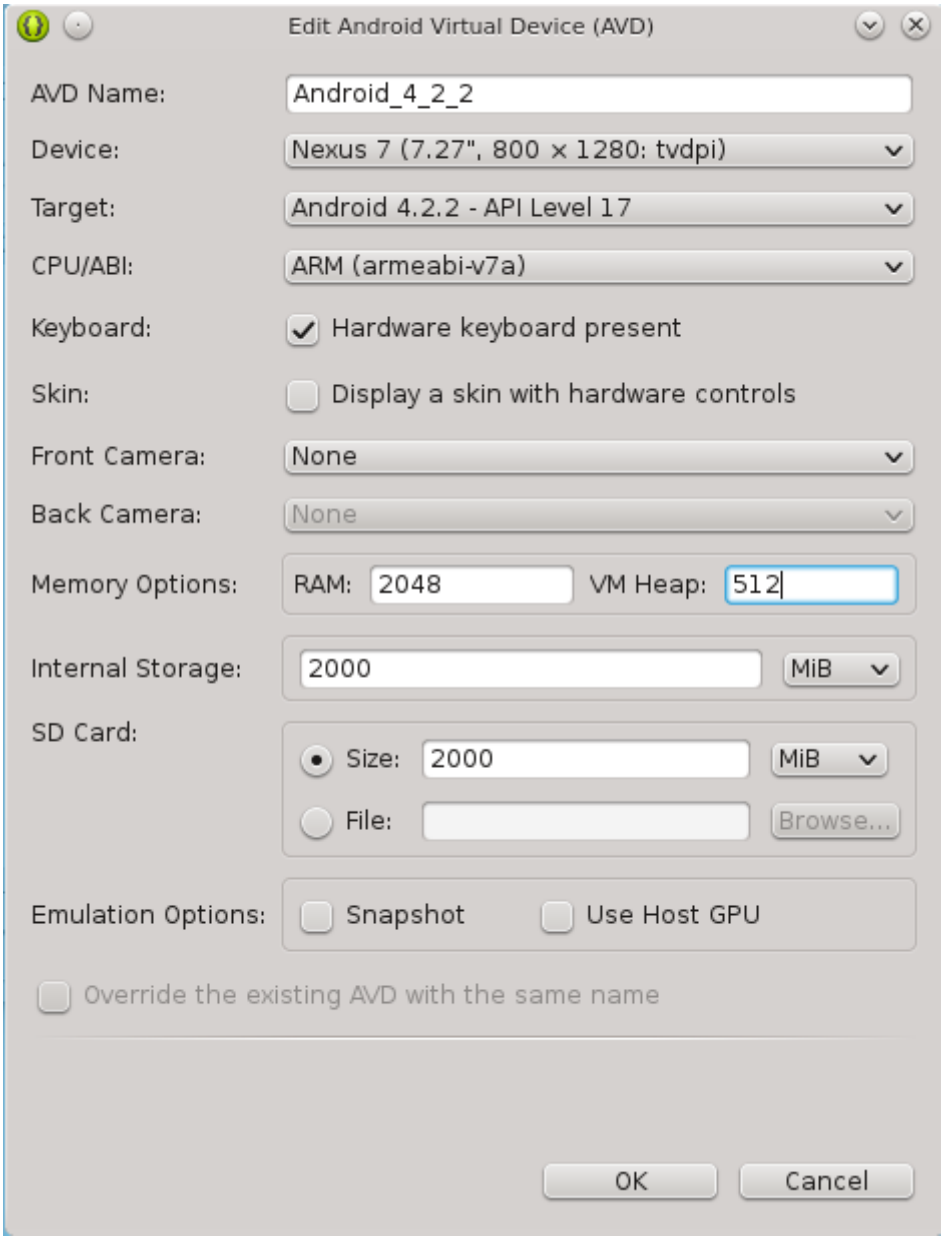

Abbildung 25 - Virtuelles Android Device anlegen

Damit wurde ein Emulator mit Android 4.2.2, einer Auflösung von 800x1280, 2GB RAM und einer Systempartition von 2GB erstellt.

Android NDK installieren

Aktuell ist das android-ndk-r9, getestet wurde mit android-ndk-r8d.

cd ~

bzip2 –d Downloads/android-ndk-r8d-linux-x86.tar.bz2

tar xvf Downloads/android-ndk-r8d-linux-x86.tar.bz2

ln –s android-ndk-r8d-linux-x86 android-ndk

Die r8d in den letzten Zeilen muss natürlich durch die verwendete Version ersetzt warden.

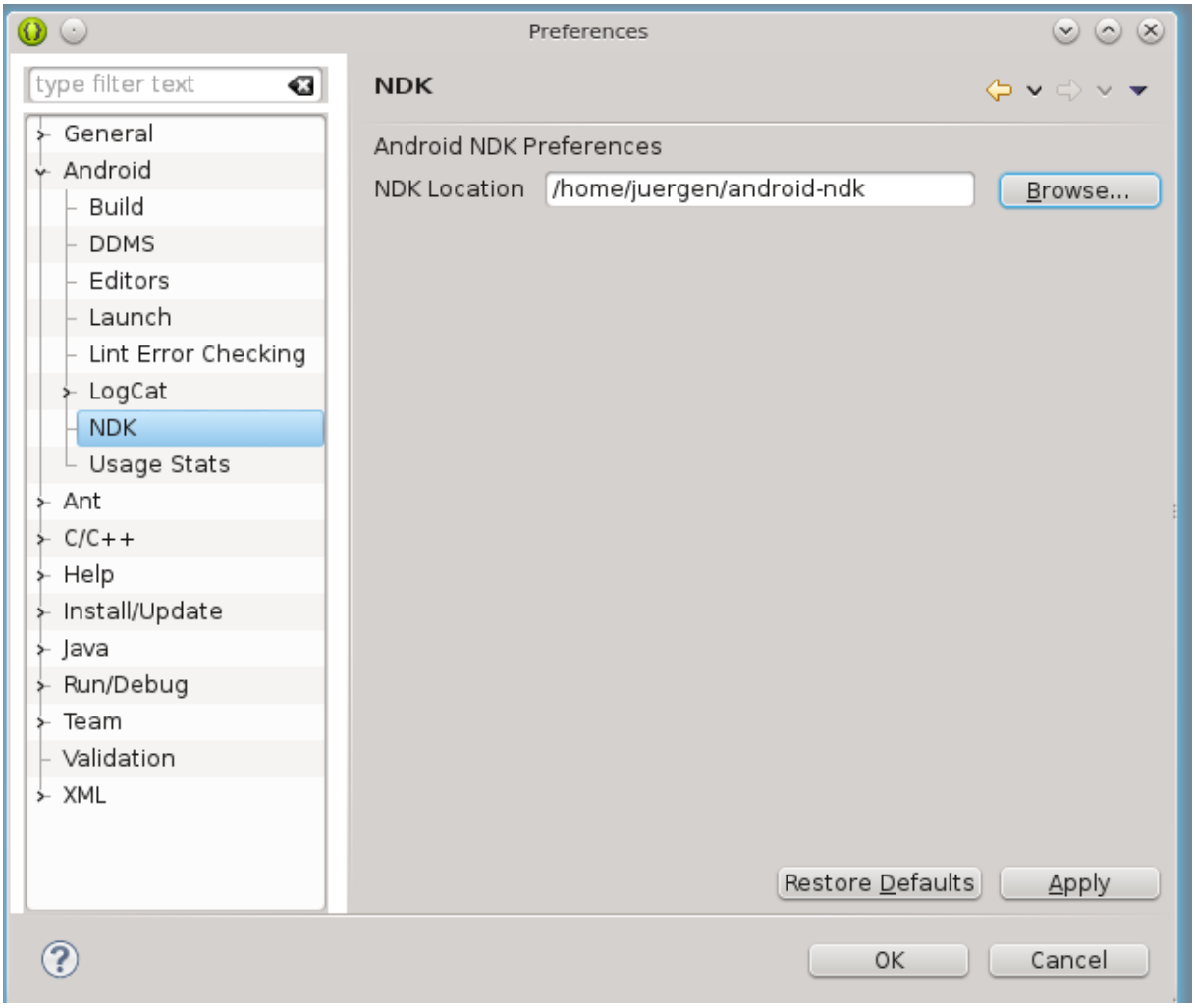

Abbildung 26 - Eintragen des NDK Pfads

Danach muss in Eclipse im Menu Window->Preferences unter dem Punkt Android->NDK der Pfad zum NDK hinterlegt werden. Dies wird benötigt, damit Eclipse die Pfade zu den Build-Tools kennt. In Abbildung 26 ist nur ein Beispiel für einen Pfad. Hier muss unbedingt der Pfad auf der eigenen Umgebung eingetragen werden: Bei mir z.B. /home/jes/android-ndk

Nun wird die eigentliche Toolchain erstellt:

cd ~/android-ndk/build/tools

./make-standalone-toolchain.sh --toolchain=arm-linux-androideabi-4.7 --arch=arm --ndk-dir=$HOME/android-ndk --system=linux-x86_64 --install-dir=$HOME/toolchain-standalone/ --platform=android-14

8. Kompilieren der einzelnen Zusatzpakete

8.1. Bison 2.7 (!)

cd ~

tar xvfz Downloads/bison-2.7.tar.gz

cd bison-2.7

Zuerst muss das Paket konfiguriert werden:
./configure --host=arm-linux --prefix=$HOME/bison CC=$HOME/toolchain-standalone/bin/arm-linux-androideabi-gcc RANLIB=$HOME/toolchain-standalone/bin/arm-linux-androideabi-ranlib AR=$HOME/toolchain-standalone/bin/arm-linux-androideabi-ar

make

make install

8.2. Flex 2.5.37

cd ~

tar xvfz flex-2.5.37.tar.gz

cd flex-2.5.37

./configure --host=arm-linux --prefix=$HOME/flex CC=~/toolchain-standalone/bin/arm-linux-androideabi-gcc RANLIB=~/toolchain-standalone/bin/arm-linux-androideabi-ranlib AR=~/toolchain-standalone/bin/arm-linux-androideabi-ar

make

make install

8.3. libgsl

cd ~

tar xvfz Downloads/gsl-1.15.tgz

cd gsl-1.15

```
./configure --host=arm-linux --prefix=$HOME/gsl CC=~/toolchain-standalone/bin/arm-linux-androideabi-gcc RANLIB=~/toolchain-standalone/bin/arm-linux-androideabi-ranlib AR=~/toolchain-standalone/bin/arm-linux-androideabi-ar --disable-shared --enable-static

make

make install
```

8.4. libjpeg

cd ~

tar xvfz Downloads/jpegsrc.v6b.tar.gz

cd jpeg-6b

./configure --host=arm-linux --prefix=$HOME/libjpeg CC=~/toolchain-standalone/bin/arm-linux-androideabi-gcc RANLIB=~/toolchain-standalone/bin/arm-linux-androideabi-ranlib AR=~/toolchain-standalone/bin/arm-linux-androideabi-ar

make

8.5. readline

cd ~

tar xvfz Downloads/readline-6.2.tar.gz

cd readline-6.2

./configure --host=arm-linux --prefix=$HOME/readline CC=~/toolchain-standalone/bin/arm-linux-androideabi-gcc RANLIB=~/toolchain-standalone/bin/arm-linux-androideabi-ranlib AR=~/toolchain-standalone/bin/arm-linux-androideabi-ar

make

make install

8.6. Berkeley DB

cd ~

tar xvfz db-5.3.21.NC.tar.gz

cd db-5.3.21.NC/build_unix

../dist/configure --host=arm-linux --prefix=$HOME/BDB CC=~/toolchain-standalone/bin/arm-linux-androideabi-gcc CXX=$HOME/toolchain-standalone/bin/arm-linux-androideabi-g++ RANLIB=~/toolchain-standalone/bin/arm-linux-androideabi-ranlib AR=~/toolchain-standalone/bin/arm-linux-androideabi-ar LD=~/toolchain-standalone/bin/arm-linux-androideabi-ld --enable-static=yes --disable-shared --enable-stl

make

make install

9. Secondo

Nun muss der Secondoquellcode geholt werden.

export CVSROOT=“:pserver:<name>@zeppelin.fernuni-hagen.de:2401/home/cvsroot“

cvs login

cvs co secondo

cd secondo/android/install

sh secondoSDK_Suse_12_2_for_android.bash

Am Schluss muss noch wie in der offiziellen Beschreibung die Zeile
source ~/.secondorc $HOME/secondo
in die Datei $HOME/.bashrc eingefügt werden.

Nun das Terminalfenster schliessen und neu öffnen

10. Secondo für Android kompilieren

Vor dem Kompilieren bitte prüfen, ob JAVA_HOME auf das JDK zeigt und nicht auf ein JRE!

cd secondo

make android

Nach dem das gesamte Programm kompiliert und gelinkt ist, befindet sich das Android Binary unter ~/secondo/android/secondoandroid/bin

Die Datei secondoandroid-debug.apk ist das komplette Archiv, das nun auf die Android Plattform installiert werden muss.

11. Installation des Programm auf der Zielplattform

cd ~/secondo/android/secondoandroid

Nun kann man z.B. den Android-Emulator z.B. aus Eclipse heraus starten

Alternativ kann der Emulator auf der Kommandozeile mit

~/android-sdk/tools/emulator @<name des AVD> gestartet werden.

In unserem Beispiel

~/android-sdk/tools/emulator @Android_4_3

Danach muss die apk-Datei auf dem Emulator installiert werden. Dies geschieht mit:

~/android-sdk/platform-tools/adb install secondoandroid-debug.apk

Für den Fall, dass das Paket schon mal installiert wurde kann man es entweder vorher entfernen oder darüber installieren.

Entfernen geschieht mit:

~/android-sdk/platform-tools/adb remove secondoandroid-debug.apk

Darüber installieren mit

~/android-sdk/platform-tools/adb install -r secondoandroid-debug.apk

Nun kann das Programm auf der Android Oberfläche gestartet werden.

12. Einfügen eigener Datenbanken

Das Programm liegt normalerweise unter /data/data/eu.ehnes.secondoandroid.

Möchte man eigene Datenbankdateien installieren und öffnen, können diese im Unterordner „files“ abgelegt werden.

Das Kommando dafür lautet:

~/android-sdk/platform-tools/adb push <Name der Datenbank> \
/data/data/eu.ehnes.secondoandroid/files

Bitte den Namen der Datenbank durch die Datei ersetzen, die man übertragen möchte.