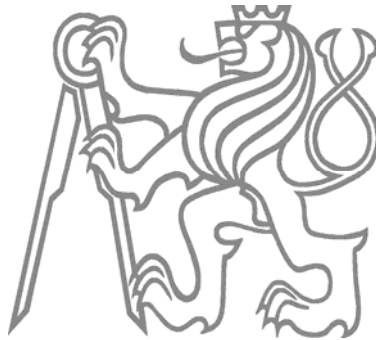


# **Semestrální projekt z předmětu X33TSW Testování softwaru**

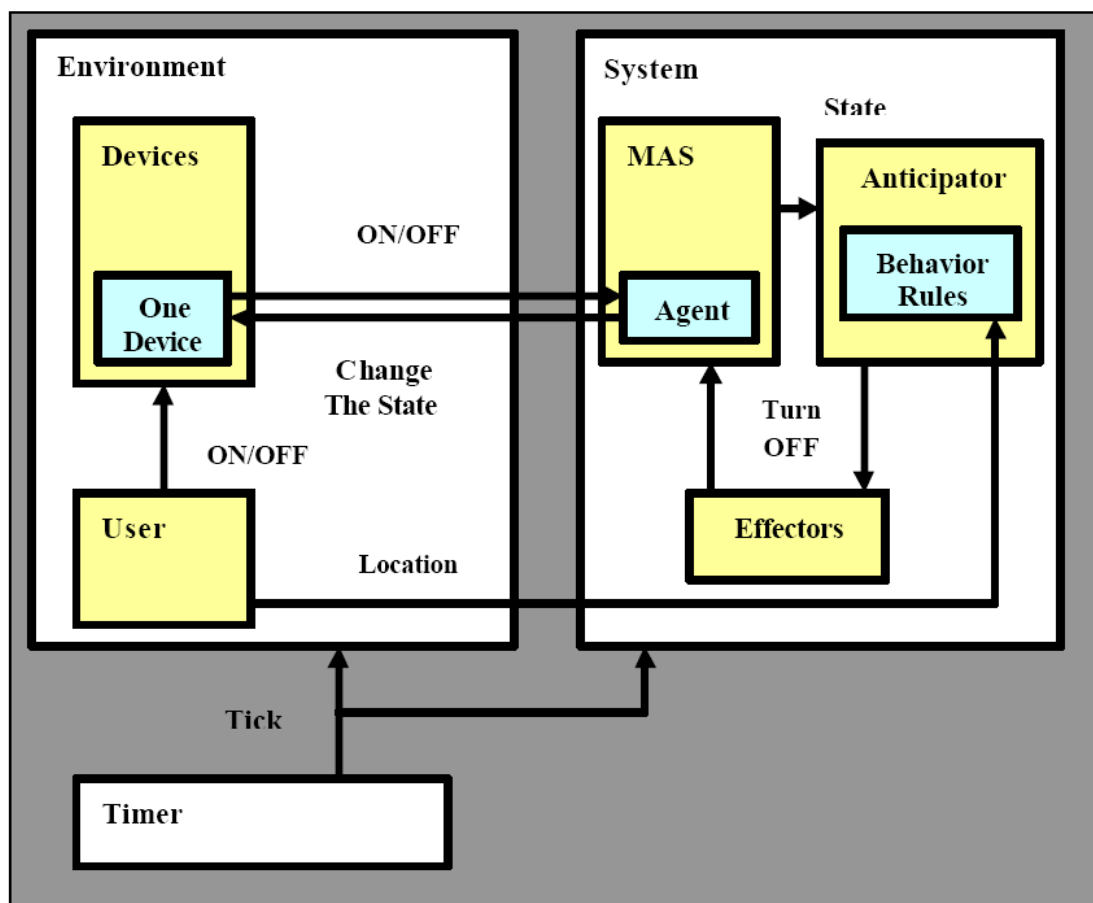


**Petr Kotál (kotalp1)**  
**Jan Ramba (rambaj1)**

## Zadání

Cílem této práce je pomocí různých nástrojů otestovat vytvořený software. Nejprve specifikujeme požadavky na software, vytvoříme UML a poté použijeme různé programy pro kontrolu správy paměti, výkonnost a efektivnost kódu. Nakonec otestujeme software pomocí manuálních i automatických testovacích programů.

Náš zvolený software je program pro testování využití anticipace pro řízení spotřeby elektrické energie v domácnosti. Jedná se o část diplomové práce na téma Anticipace a její praktické využití. Program je napsán v jazyce Java a obsahuje GUI. Následuje základní struktura programu:



## Požadavky

Pro vytvoření požadavků byl využit nástroj RequirerPro z balíku Rational Rose. Požadavky byly rozděleny na systémové (SYS), funkční (FUNC), grafické (GUI) a výsledky (RES). Následuje seznam požadavků:






Requirements	Priority	Status	Difficulty
SYS1: Run on Windows platform	Medium	Approved	Medium






Requirements	Priority	Status	Difficulty
<b>FUNC1: Create model</b> Creates model from environment.	Medium	Approved	Medium
<b>FUNC2: Perform simulation</b> Actions in one round	Medium	Approved	Medium
<b>FUNC2.1: Environment</b> Actions of environment in one round	Medium	Approved	Medium
<b>FUNC2.1.1: Generate user location</b>	Medium	Approved	Medium
<b>FUNC2.1.2: Automatic actions of devices</b>	Medium	Approved	Medium
<b>FUNC2.1.3: Generate user actions</b> User turns on and off some devices in room.	Medium	Approved	Medium
<b>FUNC2.2: Model</b> Actions of model in one round.	Medium	Approved	Medium
<b>FUNC2.2.1: Calculate actual state</b>	Medium	Approved	Medium
<b>FUNC2.2.2: Update anticipation function</b>	Medium	Approved	Medium
<b>FUNC2.2.3: Anticipate next state</b>	Medium	Approved	Medium
<b>FUNC2.2.4: Perform control actions</b>	Medium	Approved	Medium
<b>FUNC2.3: Timer</b> Generates ticks.	Medium	Approved	Medium
<b>FUNC3: Finalize simulation</b> Actions after the end of simulation.	Medium	Approved	Medium

Requirements	Priority	Status	Difficulty
<b>GUI1: Prepare simulation</b> Forms for prepare simulation. Add devices in rooms and set parameters of devices.	Medium	Approved	Medium
<b>GUI1.1: Editor of rooms</b> User can add and erase devices in rooms.	Medium	Approved	Medium
<b>GUI1.2: Editor of device properties</b>	Medium	Approved	Medium
<b>GUI2: Start simulation</b> User can set parameters of simulation.	Medium	Approved	Medium
<b>GUI3: Process simulation</b> Show states in every round.	Medium	Approved	Medium
<b>GUI4: Results</b> Show results of simulation.	Medium	Approved	Medium

Requirements	Priority	Status	Difficulty
<b>RES1: Round log</b>	Medium	Approved	Medium
<b>RES1.1: Log device change</b> Name of device, room where is device, state of device, who change the state.	Medium	Approved	Medium
<b>RES1.2: Log user position</b> In which room is user.	Medium	Approved	Medium
<b>RES1.3: Log round number</b> The number of the round.	Medium	Approved	Medium
<b>RES1.4: Log actual state</b>	Medium	Approved	Medium
<b>RES1.5: Log actual usage</b>	Medium	Approved	Medium
<b>RES1.6: Log anticipate state</b>	Medium	Approved	Medium
<b>RES2: Quality of anticipation</b>	Medium	Approved	Medium

Vzájemná závislost požadavků je vidět v traceability matici:

Relationships: - direct only	RES1: Round log	RES1.1: Log... Name of device, room where is device, state of device, who...	RES1.2: Log user... In which room is user.	RES1.3: Log round. The number of the round.
<b>FUNC1: Create model</b> Creates model from enviroment.				
▢ <b>FUNC2: Perform simulation</b> Actions in one round				
▢ <b>FUNC2.1: Enviroment</b> Actions of enviroment in one round				
<b>FUNC2.1.1: Generate user...</b>				
<b>FUNC2.1.2: Automatic...</b>				
<b>FUNC2.1.3: Generate user...</b> User turns on and off some devices in room.				
▢ <b>FUNC2.2: Model</b> Actions of model in one round.				
<b>FUNC2.2.1: Calculate...</b>				
<b>FUNC2.2.2: Update...</b>				
<b>FUNC2.2.3: Anticipate next..</b>				
<b>FUNC2.2.4: Perform control..</b>				
<b>FUNC2.3: Timer</b> Generates ticks.				
<b>FUNC3: Finalize simulation</b> Actions after the end of simulation.				

Relationships: - direct only	RES1.4: Log...	RES1.5: Log...	RES1.6: Log...	RES2: Quality of...
<b>FUNC1: Create model</b> Creates model from environment.				
<input type="checkbox"/> <b>FUNC2: Perform simulation</b> Actions in one round				
<input type="checkbox"/> <b>FUNC2.1: Environment</b> Actions of environment in one round				
<b>FUNC2.1.1: Generate user...</b>				
<b>FUNC2.1.2: Automatic...</b>				
<b>FUNC2.1.3: Generate user...</b> User turns on and off some devices in room.				
<input type="checkbox"/> <b>FUNC2.2: Model</b> Actions of model in one round.				
<b>FUNC2.2.1: Calculate...</b>				
<b>FUNC2.2.2: Update...</b>				
<b>FUNC2.2.3: Anticipate next...</b>				
<b>FUNC2.2.4: Perform control...</b>				
<b>FUNC2.3: Timer</b> Generates ticks.				
<b>FUNC3: Finalize simulation</b> Actions after the end of simulation.				

## Případy užití

Byly definovány tři případy užití:

UC1: Setting environment

User can manually prepare environment (simulation of house) using graphical interface. User must be able to select one room and change devices in this room (add new one from the list of all possible devices). He must be able to change properties of selected device.







UC2: Start simulation


User must be able to set and start simulation. He must be able to decide to use anticipation or not, start normal (show state of every round) or fast (show only result) simulation and set number of rounds.


UC3: Show results

User must be able to show results of simulation. He must be able to show record with changes in each round, show vector of usage and show the quality of anticipation.

Jejich závislost na požadavcích je vidět z traceability matice:

Relationships: - direct only	UC1: Setting enviroment	UC2: Start simulation	UC3: Show results
<b>GUI1: Prepare simulation</b> Forms for prepare simulation. Add devices in rooms and set parameters of devices.			
<b>GUI1.1: Editor of rooms</b> User can add and erase devices in rooms.			
<b>GUI1.2: Editor of device properties</b>			
<b>GUI2: Start simulation</b> User can set parameters of simulation.			
<b>GUI3: Process simulation</b> Show states in every round.			
<b>GUI4: Results</b> Show results of simulation.			

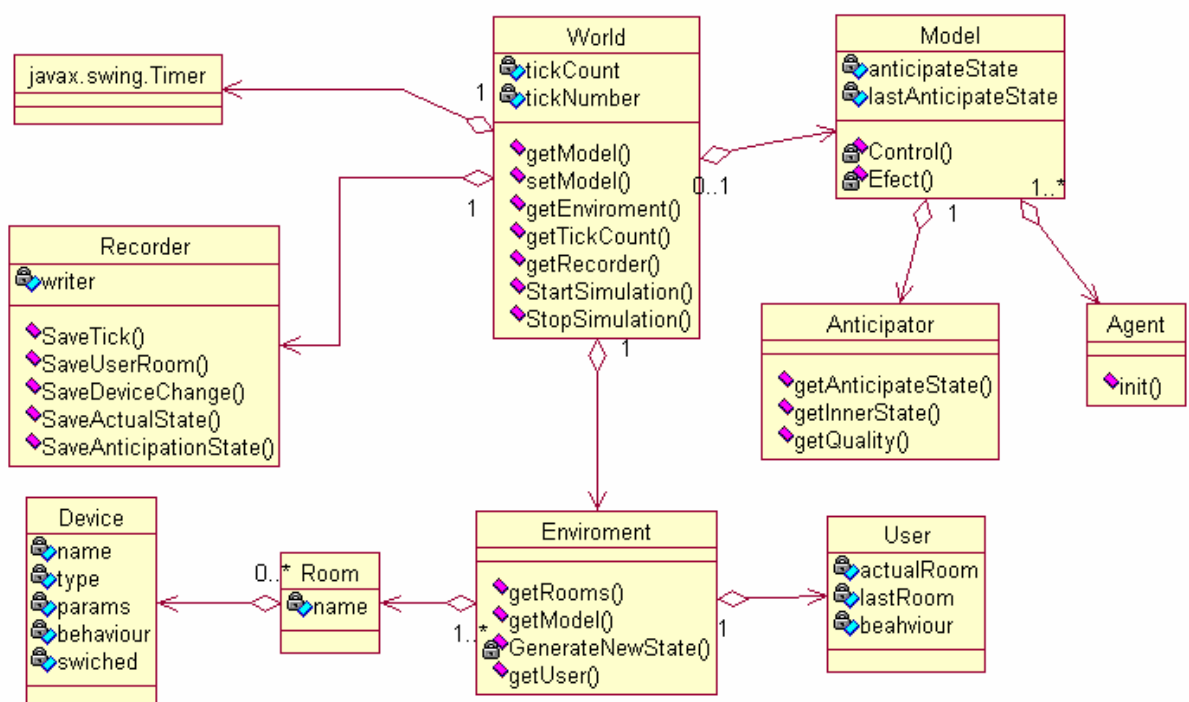
Relationships: - direct only	<b>RES1: Round log</b> RES1.1: Log device... Name of device, room where is device, state of device, who change the state.	RES1.2: Log user... In which room is user.	RES1.3: Log round... The number of the round.
<b>UC1: Setting...</b> User can manually prepare enviroment (simulation of house) using graphical interface. User must be able to select one room and change devices in this room...			
<b>UC2: Start...</b> User must be able to set and start simulation. He must be able to decide to use anticipation or not, start normal (show state of every round) or fast (show...			
<b>UC3: Show results</b> User must be able to show results of simulation. He must be able to show record with changes in each round, show vector of usages and show the quality of...			

Relationships: - direct only	RES1.4: Log actual...	RES1.5: Log actual...	RES1.6: Log...	RES2: Quality of...
<b>UC1: Setting...</b> User can manually prepare enviroment (simulation of house) using graphical interface. User must be able to select one room and change devices in this room...				
<b>UC2: Start...</b> User must be able to set and start simulation. He must be able to decide to use anticipation or not, start normal (show state of every round) or fast (show...				
<b>UC3: Show results</b> User must be able to show results of simulation. He must be able to show record with changes in each round, show vector of usages and show the quality of...				

## UML model

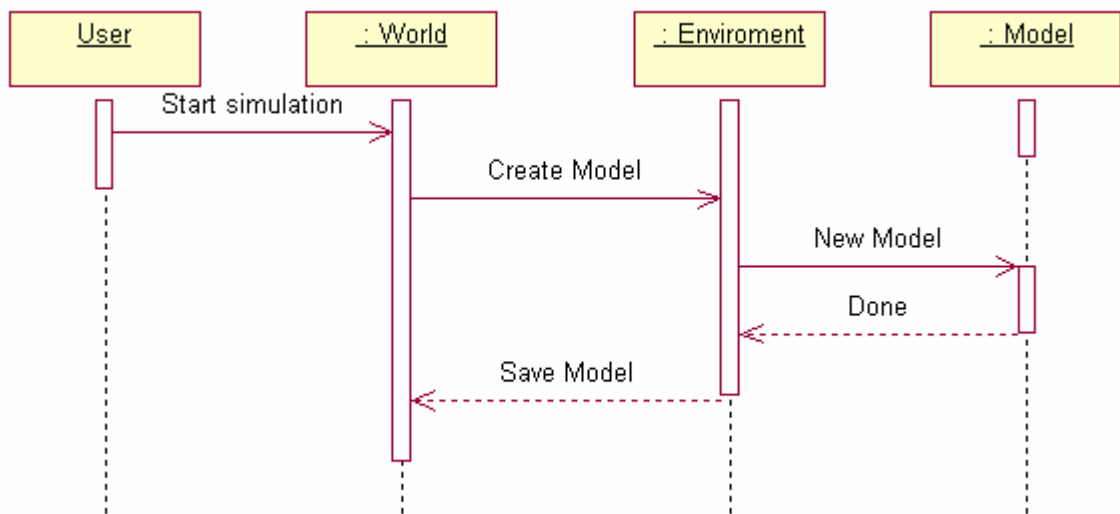
### Class diagram

Pomocí programu Rational Rose jsme vytvořili class diagram se základními třídami. Třída World obsahuje všechny ostatní třídy a poskytuje výměnu dat. Třída Enviroment představuje simulaci reálného bytu včetně uživatele v bytě. Třída Model pak vytváří model a anticipuje další vývoj. Třída Record provádí záznam všech změn během simulace. Třída Timer poskytuje diskretní čas.

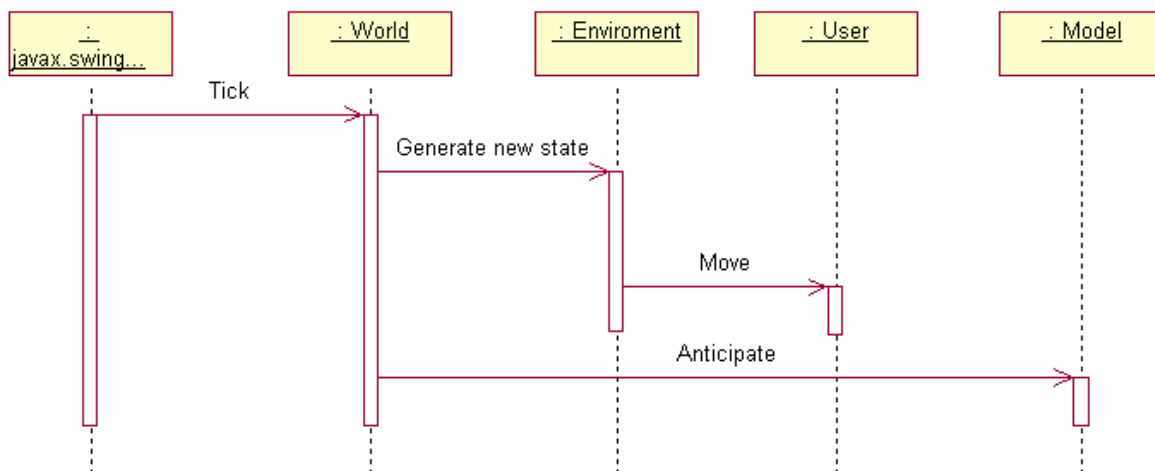


## Sequence diagram

První sequence diagram zobrazuje vytvoření modelu z prostředí po spuštění simulce.



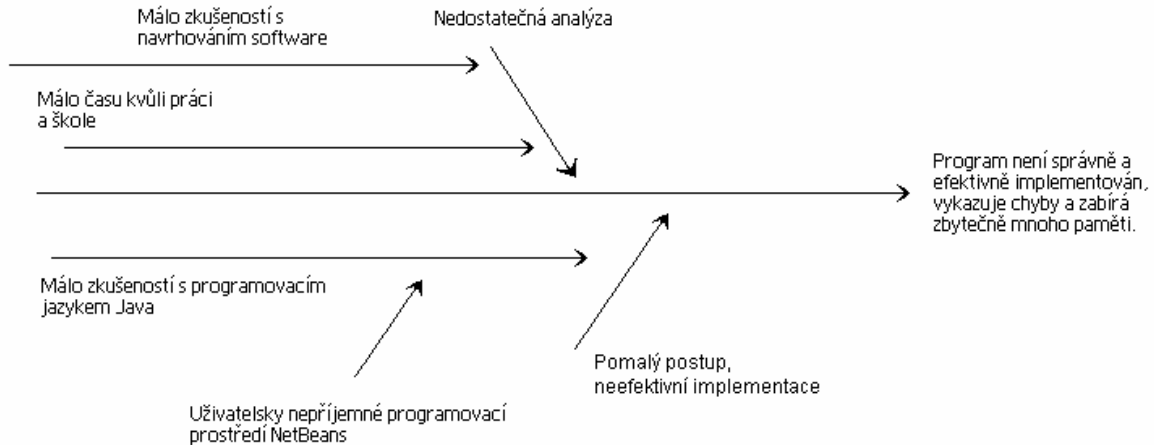
Druhý sequence diagram zobrazuje chování systému během jednoho kola. Nejprve se vygeneruje nový stav prostředí a poté se model pokouší anticipovat následující stav.





## CE diagram

Pomocí CE diagramu jsme se pokusili zapsat důvody selhání celého projektu. CE diagram umožňuje logicky zapsat výsledek brainstormingu.



## Správa paměti

Pro hledání chyb ve správě paměti jsme použili program Purify z balíku Rational Rose. Bohužel tento program nefunguje zcela ideálně pro aplikace napsané v Jave. Purify nejprve provádí tzv. instrumentaci knihoven, tedy nahrává všechny použité knihovny. Tato akce trvala více než 45 minut a Purify během ní zjistil mnoho warningů v samotných knihovnách Java i System32. Po instrumentaci se spouští samotný testovaný program, po chvíli však přestal reagovat na uživatelské akce a celý proces musel být násilně ukončen. Pro ukázkou část výpisu:

```
[E] FMM: Freeing mismatched memory in delete(void *) {1 occurrence}
Address 0x0263a4e0 points into a malloc'd block in heap 0x01740000
Location of free attempt
delete(void *) [C:\WINDOWS\system32\MSVCRT.dll]
AWTIsHeadless [C:\Program Files\Java\jdk1.5.0_12\jre\bin\awt.dll]
??? [ip=0x026482FF]
??? [ip=0x026429E3]
??? [ip=0x02642CE9]
??? [ip=0x02640217]
AsyncGetCallTrace [C:\Program Files\Java\jdk1.5.0_12\jre\bin\client\jvm.dll]
jmm_GetLastGCStat(JNIEnv_ *,_jobject *,jmmGCStat *) [C:\Program
Files\Java\jdk1.5.0_12\jre\bin\client\jvm.dll]
AsyncGetCallTrace [C:\Program Files\Java\jdk1.5.0_12\jre\bin\client\jvm.dll]
AsyncGetCallTrace [C:\Program Files\Java\jdk1.5.0_12\jre\bin\client\jvm.dll]
Allocation location
malloc [C:\WINDOWS\system32\MSVCRT.dll]
LdrLoadDll [C:\WINDOWS\system32\NTDLL.dll]
LoadLibraryExA [C:\WINDOWS\system32\kernel32.dll]
```

Místo programu Purify jsme tedy použili nástroj Profiler v NetBeans 6.0.1. Jazyk Java pro uvolňování paměti používá Garbage Collector, tedy nemá smysl zjišťovat úniky paměti.

Cílem tedy bylo nalézt objekty, které zabírají nejvíce paměti a pokusit se upravit zdrojový kód tak, aby byl efektivnější a nevyžadoval tolik paměti.

Zjistili jsme, že nejvíce paměti zabírají objekty používané v grafických knihovnách, které nelze nijak upravovat. To dokumentuje následující obrázek:

Method Name - Allocation Call Tree	Live Bytes ...	Live Bytes	Live Objects	Allocated Obje...	Avg. Age	Generations
char[]		13 392 B (100%)	197 (100%)	371	32.8	19
java.lang.String.toCharArray ()		4 152 B (31%)	2 (1%)	2	39.0	2
java.lang.String.toLowerCase (java.util.Locale)		4 024 B (30%)	90 (45,7%)	137	33.7	8
java.util.Arrays.copyOfRange (char[], int, int)		2 328 B (17,4%)	39 (19,8%)	80	40.7	10
Objects allocated by reflection		1 536 B (11,5%)	32 (16,2%)	63	18.8	4
java.lang.AbstractStringBuilder.<init> (int)		552 B (4,1%)	6 (3%)	6	41.8	4
java.util.Arrays.copyOf (char[], int)		432 B (3,2%)	6 (3%)	60	36.7	3
java.lang.CharacterData00.<clinit>		352 B (2,6%)	21 (10,7%)	21	33.0	1
java.nio.HeapCharBuffer.<init> (int, int)		16 B (0,1%)	1 (0,5%)	2	0.0	1

Po rozbalení prvního odkazu lze nalézt, že objekt se vytváří při inicializaci komponent v okně:

Method Name - Allocation Call Tree		L...	▼	Liv...	Liv...	All...	Av...	Ge...
[-]	javafx.swing.JComponent.getPreferredSize ()			30,7%	0,5%	1	33,0	1
[-]	org.netbeans.lib.awtextra.AbsoluteLayout.preferredLayoutSize (java.awt.Container)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.preferredSize ()			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.getPreferredSize ()			30,7%	0,5%	1	33,0	1
[-]	javafx.swing.JComponent.getPreferredSize ()			30,7%	0,5%	1	33,0	1
[-]	javafx.swing.JRootPane\$RootLayout.preferredLayoutSize (java.awt.Container)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.preferredSize ()			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.getPreferredSize ()			30,7%	0,5%	1	33,0	1
[-]	javafx.swing.JComponent.getPreferredSize ()			30,7%	0,5%	1	33,0	1
[-]	java.awt.BorderLayout.preferredLayoutSize (java.awt.Container)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.preferredSize ()			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.getPreferredSize ()			30,7%	0,5%	1	33,0	1
[-]	java.awt.Window.pack ()			30,7%	0,5%	1	33,0	1
[-]	powercontrol.Forms.frmResult.initComponents ()			30,7%	0,5%	1	33,0	1
[-]	powercontrol.Forms.frmResult.<init> ()			30,7%	0,5%	1	33,0	1
[-]	powercontrol.Forms.frmStartSimul.jButtonOKMouseClicked (java.awt.event.MouseEvent)			30,7%	0,5%	1	33,0	1
[-]	powercontrol.Forms.frmStartSimul.access\$000 (powercontrol.Forms.frmStartSimul, java.awt.event.MouseEvent)			30,7%	0,5%	1	33,0	1
[-]	powercontrol.Forms.frmStartSimul\$1.mouseClicked (java.awt.event.MouseEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.AWTEventMulticaster.mouseClicked (java.awt.event.MouseEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Component.processMouseEvent (java.awt.event.MouseEvent)			30,7%	0,5%	1	33,0	1
[-]	javafx.swing.JComponent.processMouseEvent (java.awt.event.MouseEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Component.processEvent (java.awt.AWTEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.processEvent (java.awt.AWTEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Component.dispatchEventImpl (java.awt.AWTEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Container.dispatchEventImpl (java.awt.AWTEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.Component.dispatchEvent (java.awt.AWTEvent)			30,7%	0,5%	1	33,0	1
[-]	java.awt.LightweightDispatcher.retargetMouseEvent (java.awt.AWTEvent)			30,7%	0,5%	1	33,0	1

Těchto případů lze najít hned několik.

Nalezli jsme pro porovnání, který objekt, vytvořený vlastním kódem, zabírá nejvíce paměti. Podle očekávání je to objekt typu Vector obsahující seznam spotřebičů.

Method Name - Allocation Call Tree		Live By...	Live B...	Live ...	Allocat...	Avg...	Gen...
java.lang.Object[]		Live By...	(100%)	(100%)	77	35.1	14
java.util.ArrayList.<init> (int)			(32,7%)	(38,8%)	27	35.3	5
java.util.Vector.<init> (int, int)			(27,8%)	(20,9%)	17	27.2	9
java.util.Vector.<init> (int)			(27,8%)	(20,9%)	17	27.2	9
java.util.Vector.<init> ()			(25,9%)	(19,4%)	16	28.5	9
powercontrol.Objects.Device.<init> (powercontrol.Objects.Room)			(9,9%)	(7,5%)	8	21.2	3
powercontrol.Objects.Device.clone ()			(6%)	(4,5%)	6	11.0	1
powercontrol.Forms.frmEditor.jButtonAddMouseClicked (java.awt.event.MouseEvent)			(4%)	(3%)	2	36.5	2
java.lang.ClassLoader.<init> (ClassLoader)			(4%)	(3%)	2	41.5	2
powercontrol.Base.ExtendedVector.<init> ()			(4%)	(3%)	2	25.0	2
java.util.zip.ZipFile.<init> (java.io.File, int)			(2%)	(1,5%)	1	43.0	1
sun.awt.im.ExecutableInputMethodManager.<init> ()			(2%)	(1,5%)	1	40.0	1
java.awt.Window.<init> ()			(2%)	(1,5%)	1	37.0	1
java.awt.Window.<init> (java.awt.GraphicsConfiguration)			(2%)	(1,5%)	1	11.0	1
javafx.swing.text.DefaultHighlighter\$SafeDamager.<init> (javafx.swing.text.DefaultHighlighter)			(2%)	(1,5%)	1	11.0	1
javafx.swing.plaf.metal.MetalLookAndFeel.initComponentDefaults (javafx.swing.UIManager)			(15,6%)	(6%)	5	44.0	1
javafx.swing.ArrayTable.put (Object, Object)			(9,9%)	(16,4%)	11	40.6	2
javafx.swing.event.EventListenerList.add (Class, java.util.EventListener)			(7,1%)	(9%)	11	29.8	3
java.util.Arrays.copyOf (Object[], int, Class)			(2,8%)	(4,5%)	3	40.0	1
sun.awt.util.IdentityArrayList.<init> (int)			(2%)	(1,5%)	1	46.0	1
Objects allocated by reflection			(1,1%)	(1,5%)	1	44.0	1
javafx.swing.text.StyleContext\$SmallAttributeSet.<init> (javafx.swing.text.StyleContext, javafx.swing.text.AttributeSet)			(0,9%)	(1,5%)	1	41.0	1

Tento objekt však již nelze nijak optimalizovat a při porovnání s objekty, které používá grafické rozhraní, je jeho paměťová náročnost o mnoho menší.

## Výkonnost kódu

Podle zadání jsme měli použít program Quantify, který však pro aplikace napsané v Java nefunguje. Použili jsme tedy nástroj Profiler obsažený v NetBeans 6.0.1, který nám změřil dobu trvání jednotlivých funkcí při běhu programu.

Call Tree - Method		Time [%]	Time	Invocations
powercontrol.Forms.frmEditor.access\$400	(powercontrol.Forms.frmEditor, java.awt.event.MouseEvent)		1618 ms (63,9%)	2
powercontrol.Forms.frmEditor.jButtonStartMouseClicked	(java.awt.event.MouseEvent)		1618 ms (63,9%)	2
powercontrol.Forms.frmStartSimul\$1.mouseClicked	(java.awt.event.MouseEvent)		1454 ms (57,5%)	2
powercontrol.Forms.frmStartSimul.access\$000	(powercontrol.Forms.frmStartSimul, java.awt.event.MouseEvent)		1454 ms (57,5%)	2
powercontrol.Forms.frmStartSimul.jButtonOKMouseClicked	(java.awt.event.MouseEvent)		1454 ms (57,5%)	2
powercontrol.World.StartSimulation	(boolean, int)		1265 ms (50%)	2
powercontrol.World.Tick	()		1229 ms (48,6%)	2002
powercontrol.Model\$1.tick	(powercontrol.Objects.TickEvent)		638 ms (25,2%)	2002
powercontrol.Environment\$1.tick	(powercontrol.Objects.TickEvent)		556 ms (22%)	2002
powercontrol.Environment.access\$000	(powercontrol.Environment)		554 ms (21,9%)	2002
powercontrol.Environment.GenerateNewState	()		553 ms (21,9%)	2002
powercontrol.Objects.Device.changeSwitched	(boolean)		187 ms (7,4%)	10901
powercontrol.Objects.Device.change	()		149 ms (5,9%)	10440
powercontrol.Objects.Agent\$1.SwitchChanged	(java.util.EventObject)		140 ms (5,5%)	10440
powercontrol.Objects.Agent.setDeviceSwitched	(String, boolean)		133 ms (5,3%)	10440
powercontrol.Objects.Vectors.DeviceVector.ExistByName	(String)		57,1 ms (2,3%)	10440
powercontrol.Objects.Vectors.DeviceVector.FindByName	(String)		54,2 ms (2,1%)	10440
Self time			9,50 ms (0,4%)	10440
powercontrol.Objects.Device.setSwitched	(boolean)		1,72 ms (0,1%)	10440
Self time			3,87 ms (0,2%)	10440
Self time			5,78 ms (0,2%)	10440
Self time			14,8 ms (0,6%)	10901
powercontrol.Objects.User.getActualRoom	()		1,60 ms (0,1%)	10901
powercontrol.World.getEnvironment	()		1,4 ms (0%)	10901
powercontrol.World.getTickCount	()		0,942 ms (0%)	10901
powercontrol.Objects.Behaviors.UserControl.canChange	(powercontrol.Objects.Room)		0,807 ms (0%)	8148
powercontrol.Objects.Behaviors.UserDuration.canChange	(powercontrol.Objects.Room)		0,260 ms (0%)	1371

Po zhodnocení získaných údajů jsem našel část kódu, která nebyla optimální a způsobovala snížení výkonnosti programu. Jedná se o vyhledávání modelu spotřebiče při řízené změně stavu, které lze změnou v implementaci řízení zcela vypustit.

Další neoptimální části kódu jsme již nenašli.

## Manuální testování

Tento úkol nemohl být splněn, protože se nám nepovedlo spustit program Rational Manual Tester, respektive nepodařilo se nám založit projekt, který by v tomto programu mohl být otevřen. Po prostudování dostupných podkladů jsme zjistili, že se jedná o program umožňující zadávání, správu a vyhodnocování testů programu, které se provádějí ručně. Nejprve se nový manuální test přiřadí některému již existujícímu testovacímu případu (test case) a poté podrobně rozepíše po jednotlivých krocích. Lze určit kroky, které představují verifikační body, jejich cílem je členění testu na více částí a jejich snadnější vyhodnocení. Takto nakonfigurovaný test je potom testerem podle instrukcí proveden a tester manuálně nastaví výsledky jednotlivých kroků.

## Automatické testování grafického rozhraní

Cílem tohoto úkolu je použít program Rational Robot pro vytvoření automatických testů grafického rozhraní. Program Rational Robot umožňuje vytvářet skripty, které představují jednotlivé testy. Tyto skripty umožňují v pseudojazyce, který vychází z Visual Basic, naprogramovat, co má být vyplněno do zadávacích políček v GUI, která tlačítka mají být kliknuta, jaká položka se má vybrat z výběrového menu apod. Opět je možné určit

verifikační body, které mají stejný význam jako v předcházející kapitole. Skript se poté v programu Rational Robot spustí a provede zadaný test. Na konci testu jsou zobrazeny výsledky, pokud například nastala chyba a byla zobrazena v okně, je toto okno uloženo jako screenshot.

Při vytváření skriptu se nám nepovedlo nalézt příkazy, které by fungovaly s testovaným programem napsaným v jazyce Java. Rational Robot naštěstí umožňuje vytvoření skriptu, který zaznamenává pohyb a akce kurzoru na obrazovce, lze tedy vytvořit alespoň takto jednoduchý test. Pochopitelně nelze určit verifikační body.

Testovací skript:

Sub Main

Dim Result As Integer

'Initially Recorded: 30.4.2008 17:12:44

'Script Name: TEST01

'Window SetContext, "Caption=dist", ""

'ListView DblClick, "Text=FolderView;\;ItemText=Power\_Control", "Coords=100,18"

StartApplication ""G:\X33TSW\Power\_Control-java\dist\Power\_Control.bat""

Window SetContext, "Caption=Editor of rooms in house", ""

'JavaListView Click, "Name=jListTypes;\;ItemText=Fridge", ""

'PushButton Click, "Name=jButtonAdd"

Window Click, "", "Coords=378,99"

Window Right\_Click, "", "Coords=291,182"

Window Click, "", "Coords=379,119"

Window Click, "", "Coords=301,177"

Window Click, "", "Coords=367,139"

Window DblClick, "", "Coords=289,180"

Window Click, "", "Coords=93,116"

Window Click, "", "Coords=292,223"

Window Click, "", "Coords=369,171"

Window DblClick, "", "Coords=287,176"

Window Click, "", "Coords=242,45"

Window Click, "", "Coords=208,86"

Window Click, "", "Coords=376,98"

Window Click, "", "Coords=279,176"

Window Click, "", "Coords=374,112"

Window DblClick, "", "Coords=280,181"

Window Click, "", "Coords=370,169"

Window Click, "", "Coords=282,174"

Window Click, "", "Coords=370,225"

Window Click, "", "Coords=302,171"

Window Click, "", "Coords=347,239"

Window Click, "", "Coords=296,173"

Window Click, "", "Coords=245,42"

Window Click, "", "Coords=214,108"

Window Click, "", "Coords=353,260"

Window DblClick, "", "Coords=286,174"  
Window Click, "", "Coords=286,174"  
Window Click, "", "Coords=359,293"  
Window Click, "", "Coords=301,178"  
Window Click, "", "Coords=356,310"  
Window Click, "", "Coords=299,176"  
Window Click, "", "Coords=365,187"  
Window Click, "", "Coords=311,175"  
Window Click, "", "Coords=349,225"  
Window Click, "", "Coords=300,175"  
Window Click, "", "Coords=243,41"  
Window Click, "", "Coords=186,126"  
Window Click, "", "Coords=383,187"  
Window Click, "", "Coords=304,175"  
Window Click, "", "Coords=360,114"  
Window Click, "", "Coords=285,178"  
Window Click, "", "Coords=355,260"  
Window DblClick, "", "Coords=304,184"  
Window Click, "", "Coords=105,136"  
Window Click, "", "Coords=40,410"

Window SetContext, "Caption=Editor of device properties", ""  
InputKeys "2"  
Window Click, "", "Coords=72,77"  
InputKeys "{DELETE}8"  
Window Click, "", "Coords=22,101"  
Window Click, "", "Coords=35,152"

Window SetContext, "Caption=Editor of rooms in house", ""  
Window Click, "", "Coords=247,45"  
Window Click, "", "Coords=194,87"  
Window Click, "", "Coords=112,120"  
Window Click, "", "Coords=51,406"

Window SetContext, "Caption=Editor of device properties", ""  
InputKeys "8"  
Window Click, "", "Coords=73,75"  
InputKeys "{BKSP}{DELETE}{DELETE}98"  
Window Click, "", "Coords=22,100"  
Window Click, "", "Coords=48,146"

Window SetContext, "Caption=Editor of rooms in house", ""  
Window Click, "", "Coords=486,46"

Window SetContext, "Caption=Simulation params", ""  
Window Click, "", "Coords=21,18"  
Window Click, "", "Coords=123,41"  
InputKeys "{BKSP}10"  
Window Click, "", "Coords=51,119"

Window SetContext, "Caption=Result", ""  
 Window MoveTo, "", "Coords=255,239"  
 Window CloseWin, "", ""

Window SetContext, "Caption=Editor of rooms in house", ""  
 Window CloseWin, "", ""

End Sub

Výsledky testu:

Event Type	Result	Date & Time	Failure Reason	Computer Name	Defects
Computer Start	Pass	28.5.2008 16:18:07		132-09	
Script Start (TEST01)	Pass	28.5.2008 16:18:07		132-09	
Application Start	Pass	28.5.2008 16:18:07		132-09	
Script End (TEST01)	Pass	28.5.2008 16:18:33		132-09	
Computer End	Pass	28.5.2008 16:18:33		132-09	

## Propagace kovarianční matice

Testovaný software neobsahuje vhodnou funkci pro výpočet propagace kovarianční matice, proto byla s ohledem na výpočetní náročnost vybrána následující funkce jako příklad:

$$y = \sin x^3 + q$$

### Explicitní propagace

$$q = y - \sin x^3$$

$$J = \begin{vmatrix} \frac{\partial q}{\partial x} & \frac{\partial q}{\partial y} \end{vmatrix} = \begin{vmatrix} -3x^2 \cos x^3 & 1 \end{vmatrix}$$

$$\Sigma_{qq} = |\sigma_q|^2 = J \begin{vmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{vmatrix} J^{-1} = \begin{vmatrix} -3x^2 \cos x^3 & 1 \end{vmatrix} \begin{vmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{vmatrix} \begin{vmatrix} -3x^2 \cos x^3 & 1 \end{vmatrix}$$

$$= 9x^4 \cos^2 x^3 \sigma_x^2 - 6x^2 \cos x^3 \sigma_{xy} + \sigma_y^2$$

### Implicitní propagace

$$F(x, y, q) = (y - \sin x^3 - q)^2$$

$$f = \frac{\partial F}{\partial q} = -2(y - \sin x^3 - q)$$

$$\frac{\partial f}{\partial q} = 2 \quad \frac{\partial f}{\partial x} = 6x^2 \cos x^3 \quad \frac{\partial f}{\partial y} = -2$$

$$\Sigma_{qq} = |\sigma_q|^2 = \left| \frac{\partial f}{\partial q} \right|^{-1} \begin{vmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{vmatrix} \begin{vmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{vmatrix} \begin{vmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{vmatrix} \left| \frac{\partial f}{\partial q} \right|^{-1} = \begin{vmatrix} 3x^2 \cos x^3 & -1 \end{vmatrix} \begin{vmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{vmatrix} \begin{vmatrix} 3x^2 \cos x^3 \\ -1 \end{vmatrix}$$

$$= 9x^4 \cos^2 x^3 \sigma_x^2 - 6x^2 \cos x^3 \sigma_{xy} + \sigma_y^2$$

Z vypočtených výsledků kovarianční matice je vidět, že chyby na vstupu mají značný vliv na výstup. Ten je dán hlavně první členem  $9x^4 \cos^2 x^3 \sigma_x^2$ .

## Závěr

Cílem této práce bylo seznámení se základními nástroji používanými při navrhování a testování softwaru. Abychom si tyto nástroje odzkoušeli v praxi, použili jsme program Power Control napsaný v jazyce Java, a pokusili se ho otestovat. Od začátku jsme se potýkali se dvěma problémy.

Doporučené programy měly veliké problémy s použitým jazykem Java, v několika případech nám nevrátili žádné výsledky, nebo nedokázali program Power Control vůbec otestovat. Pokusili jsme se najít náhradní řešení, které částečně poskytl program NetBeans 6.0.1 svým nástrojem Profiler.

Další problém byl nakonfigurování a spuštění samotných programů pro testování, zvláště pak programů určených pro manuální a automatické testování, kdy se nám program Rational Manual Tester vůbec nepodařilo spustit. Vinu na tom zřejmě nese špatná instalace těchto programů na některých počítačích v učebně K132.

I přes tyto obtíže se nám povedlo většinu úkolů splnit a v odzkoušeli jsme si tak postupy pro testování softwaru. Bohužel doporučené programy zřejmě nejsou moc vhodné pro testování programů napsaných v jazyce Java.