

Building Software using Rich Clients Platforms

Rikard Thulin

Speakers Profile

Senior Consultant & Java Evangelist

Co-leader for Gothenburg's Java User
Group, Javaforum

rikard (at) thulin.pp.se

Agenda

- The needs of an advanced rich client
- Why we need Application Architecture
- Available Rich Client Frameworks
- Demo
- Real world experience
- Conclusions
- Q & A

Local vs Web application

	Local app (RCP based)	Web app
Deployment	Automatic updates, clients can have difference versions	Clients are always updated and in sync
UI Components	Very rich and mature	Fancy but not mature, usability is low
Performance	Very high, latency problems can be reduced	Slow
Time to develop	Depends on skill set	Depends on skill set
Market	Small	The whole world

The needs

- Time-to-market
- Focus on business logic
- Usability
- Flexible deployment and distribution
- Solve the application architecture problem

Time-to-market

- Requires initial investment
- Hurts TTM in short term
- Huge decrease in mid to long term
- RCP might not be the best choice for simple applications

Focus on business logic

What the heck is this?

Focus on business logic

- Today - no one needs to write their own web framework
- Today - no one needs to not write their own Rich Client application framework
- Instead of doing lots (and lots) of UI infrastructure - do real business value

Focus on business logic

- Most applications need some kind of infrastructure to handle things such as
 - actions, menus, keyboard shortcuts, window management, state, etc
- Swing/SWT does not address this - at the end of the day they are just widget libraries

Focus on business logic

- The RCP allows you to focus on the business - not the UI infrastructure
- True also for web applications as well (no one writes plain JSP)

Usability

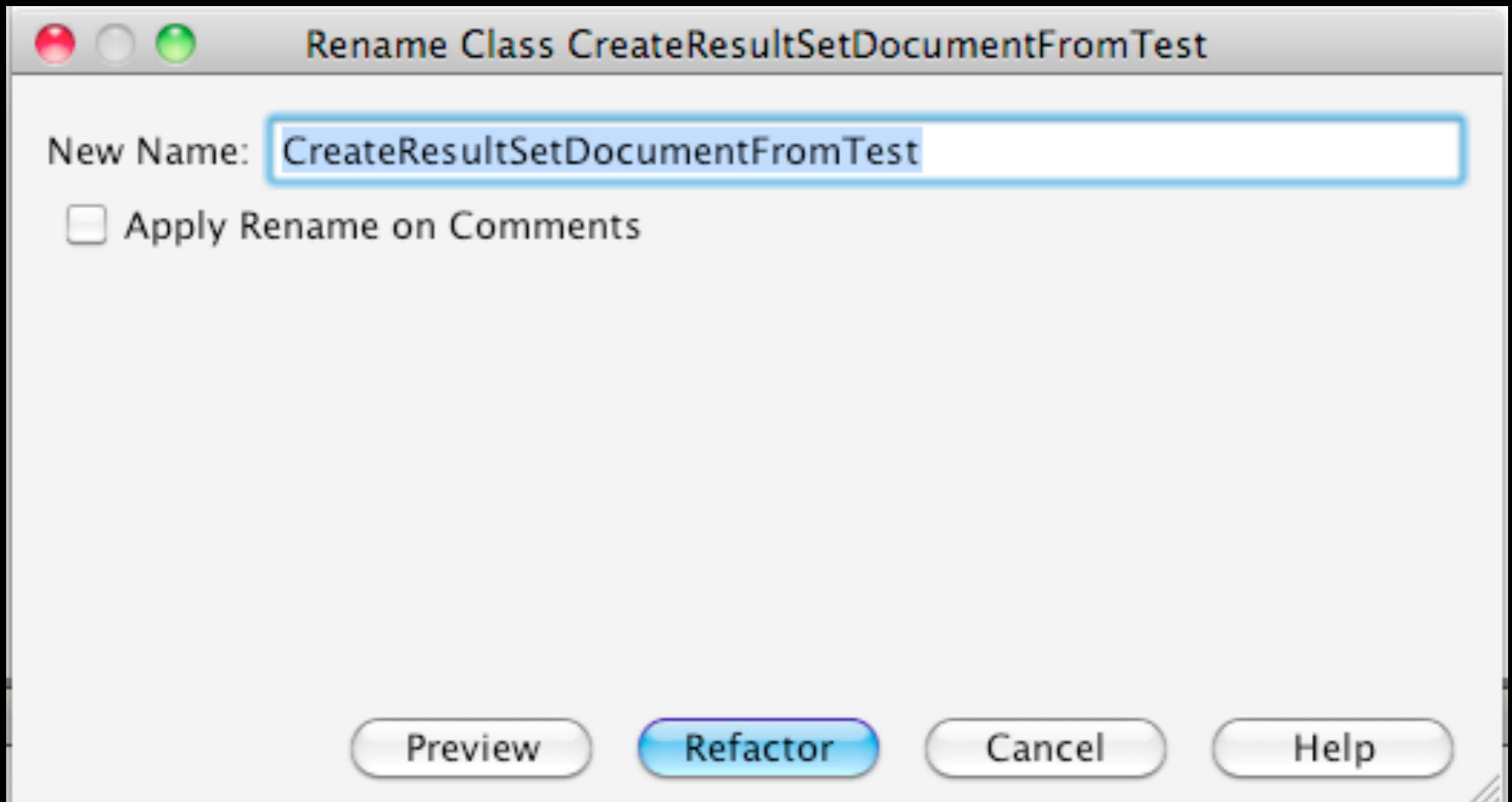
- Application level usability comes for free
- Man years of usability out-of-the box
- The RCP helps you with lower level usability as well
- Lower level usability is pretty much up to you to get right

Usability

- To demonstrate usability
 - First an RCP based dialog will be presented
 - Second a non RCP based dialog will be presented
- Try to figure out the five usability things addressed by the RCP version

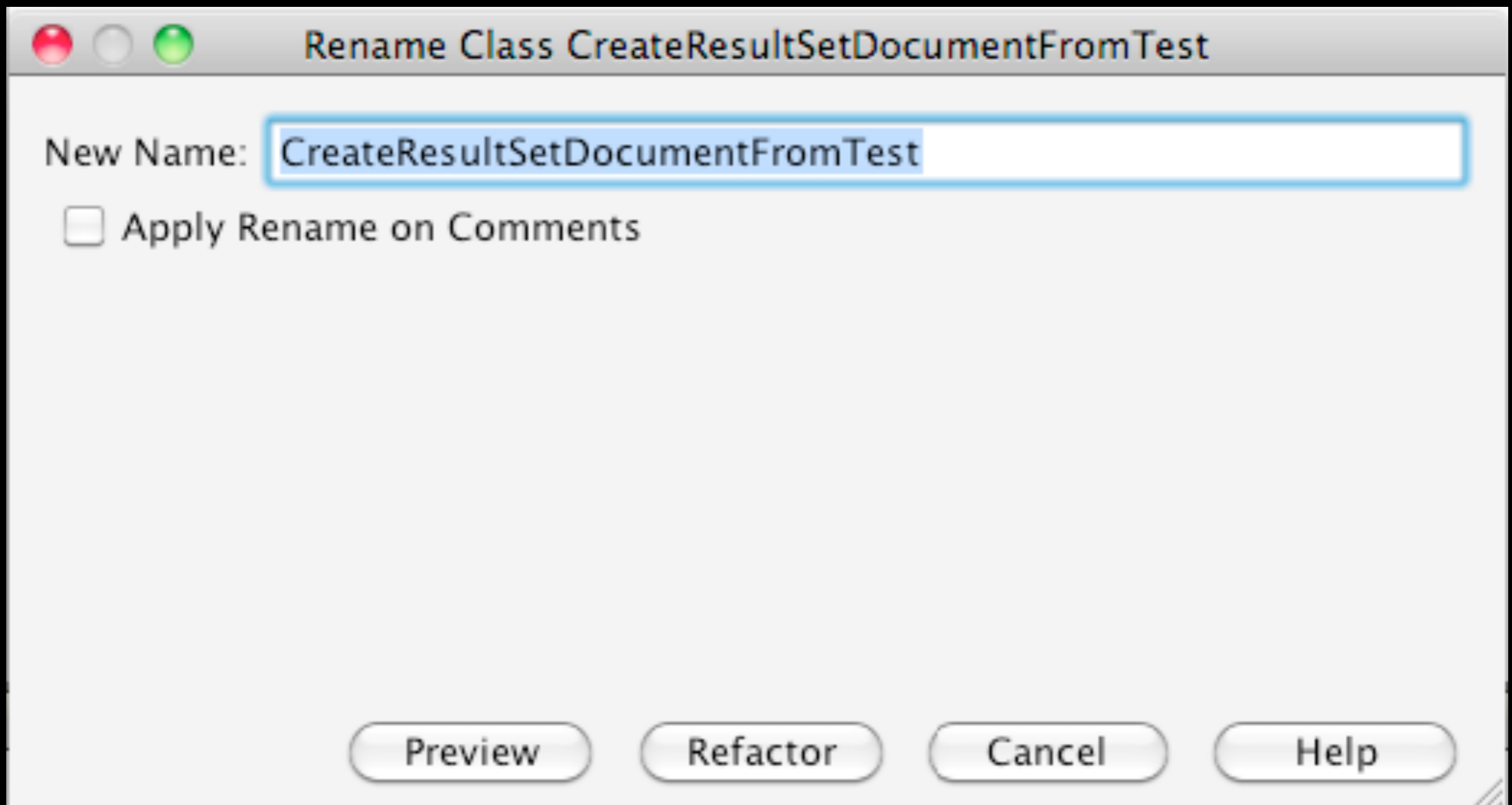
Usability

Dialog implemented using an RCP



Usability

The same dialog implemented without an RCP

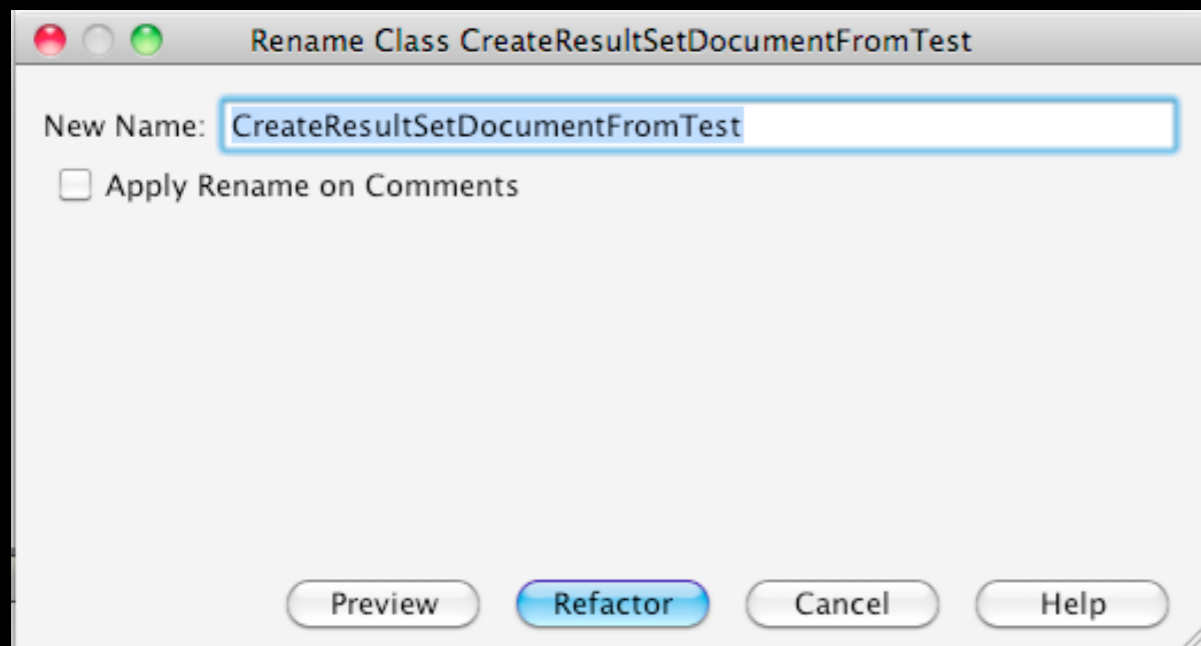


Usability

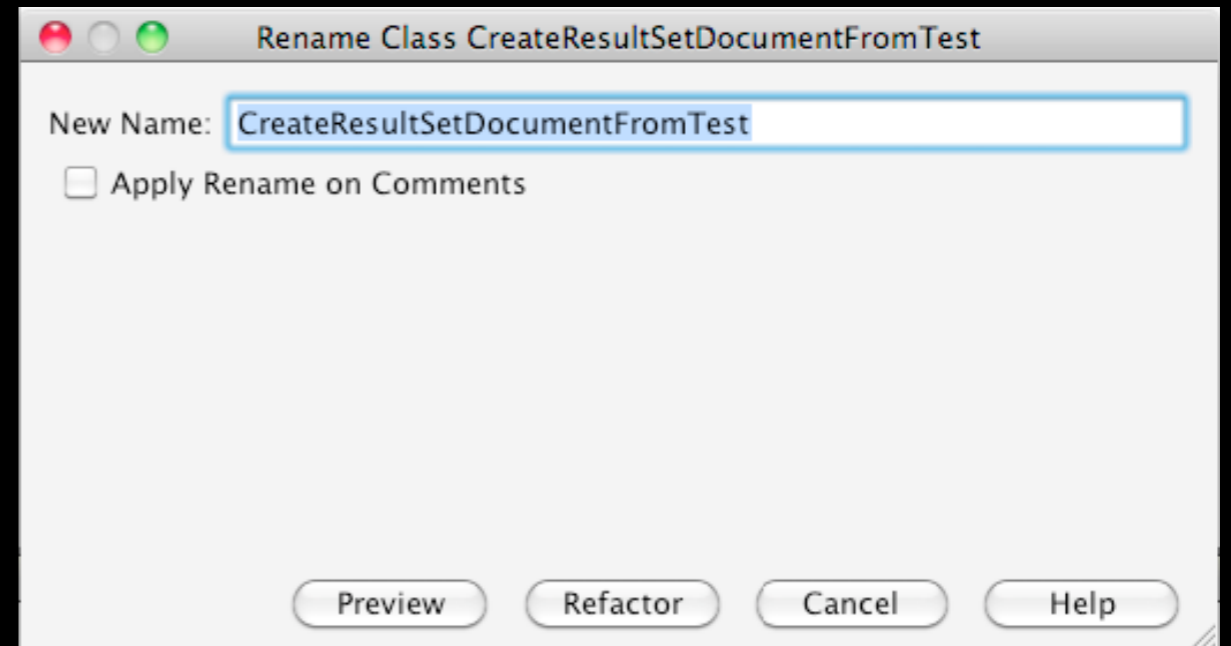
Did you find five
usability issues addressed
by the RCP?

Usability

Side by side comparison

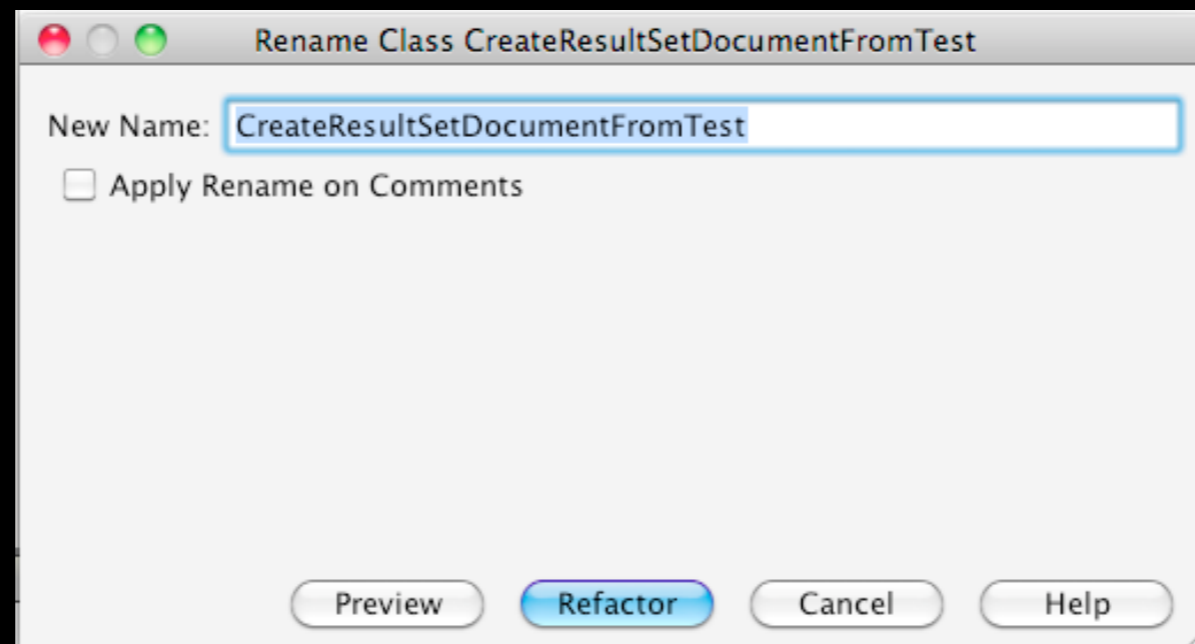


RCP



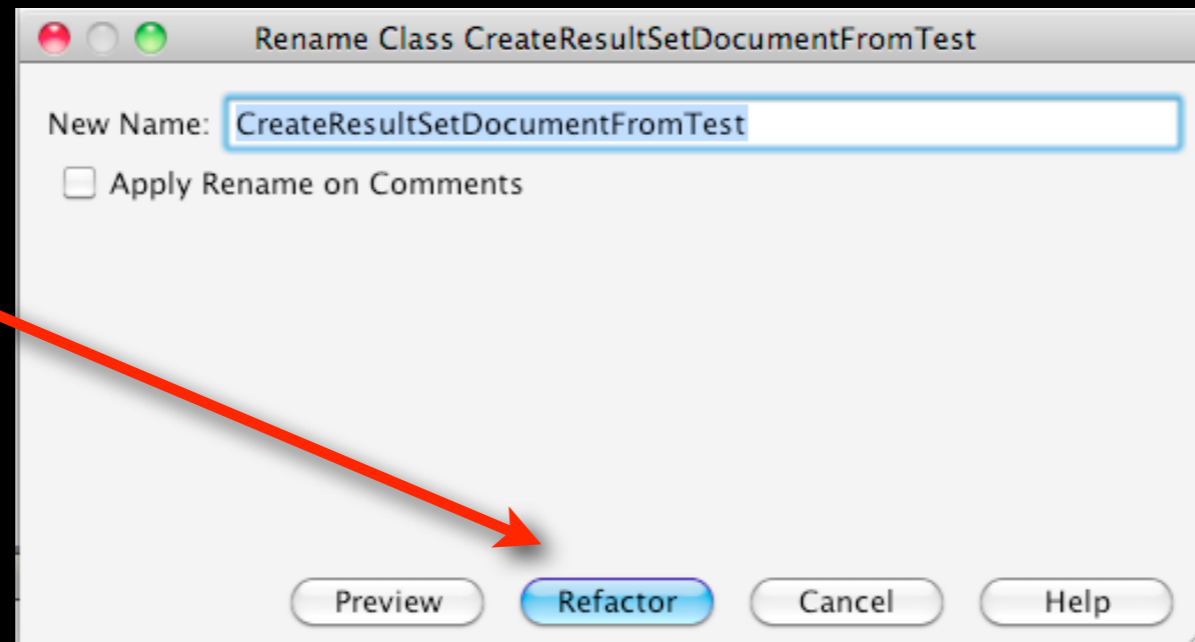
Non RCP

Usability



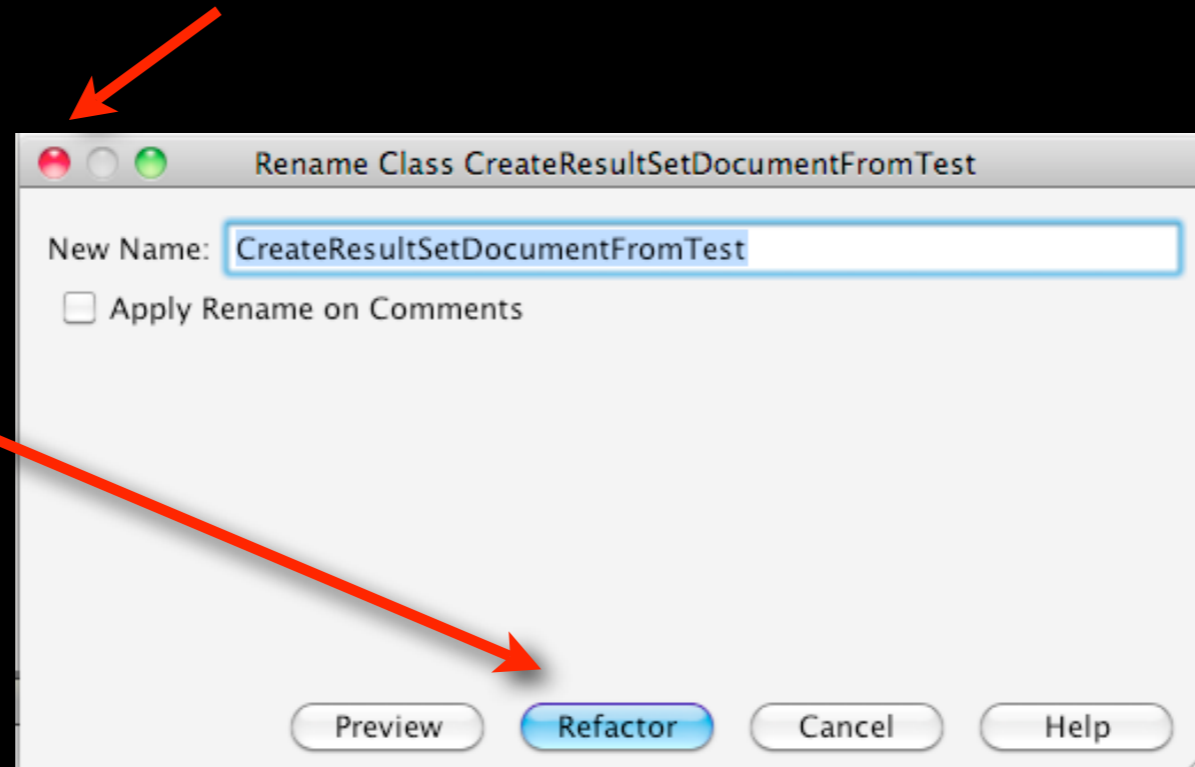
Usability

Default button



Usability

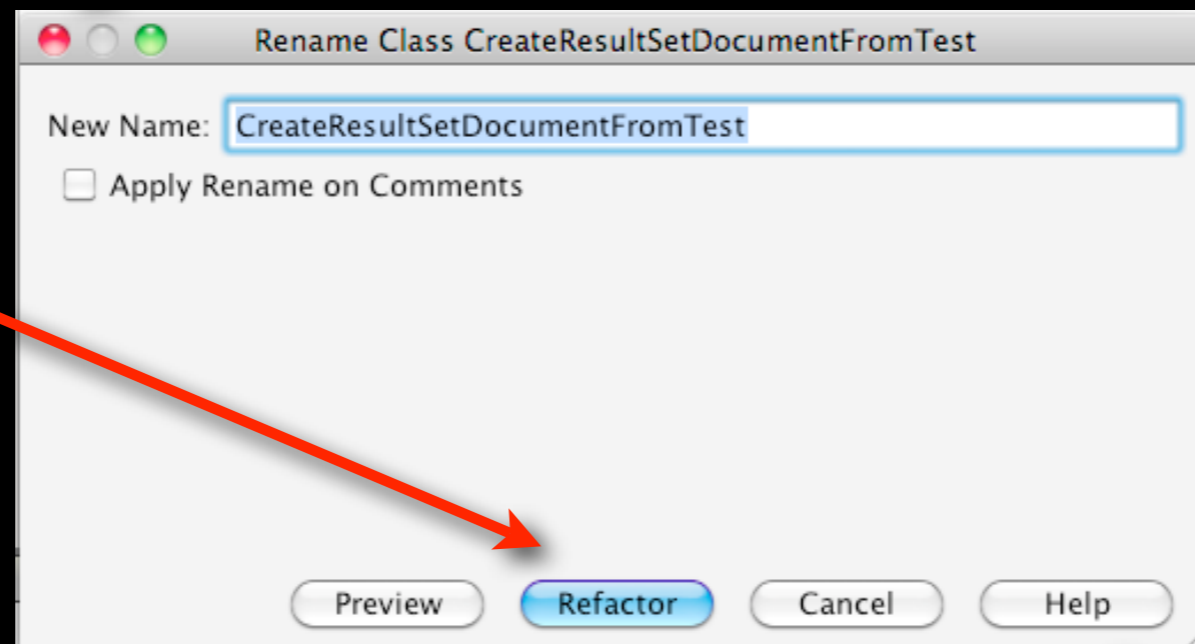
Escape closes the dialog



Default button

Usability

Escape closes the dialog

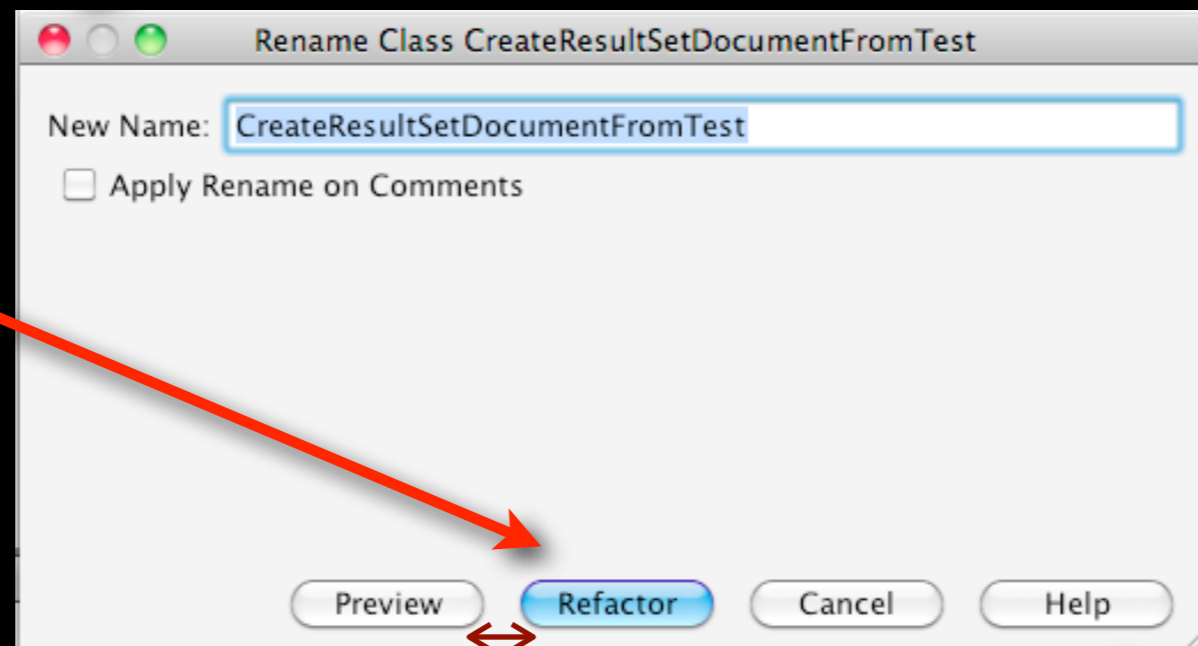


Default button

Build in support for help

Usability

Escape closes the dialog



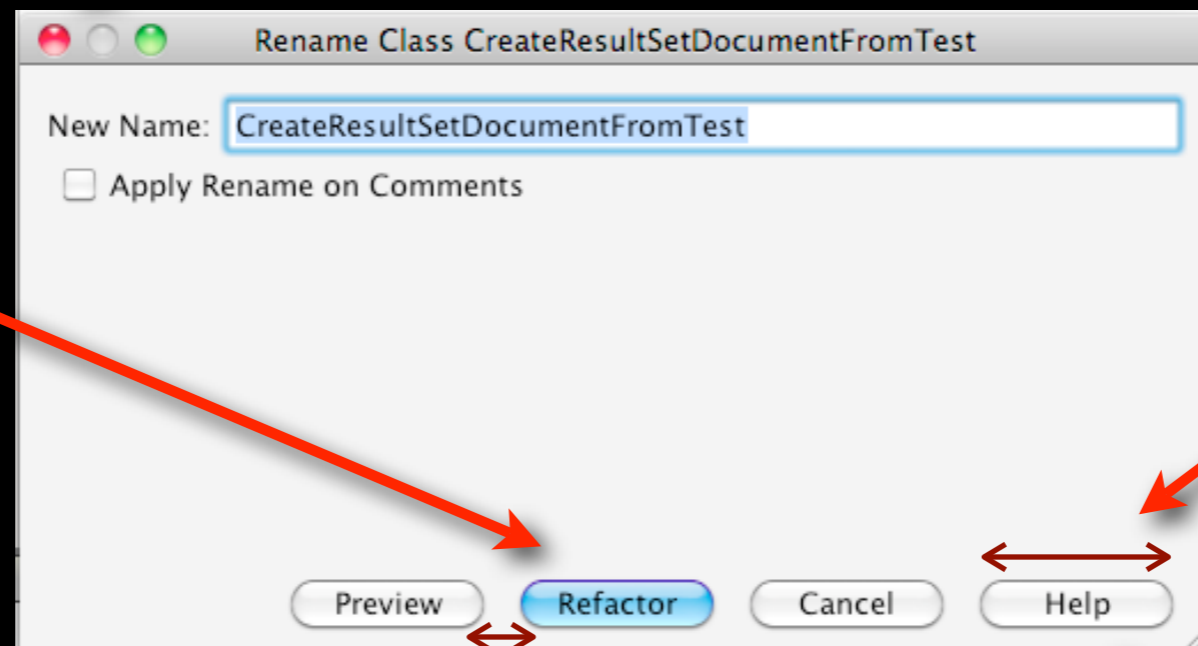
Default button

Correct spacing

Build in support for help

Usability

Escape closes the dialog



Default button

Correct size

Correct spacing

Build in support for help

Application architecture

- An application is divided into modules/
plugins/bundles
- Plugins declares their dependencies
- Plugins publish APIs and hide internal code
- Plugins are loosely coupled
- Plugins execute in a runtime container

Application architecture

- A modular system does not prevent bad coding practices
- But it encourages good coding practices
- Bad code is isolated to one modules and not spread across the whole application

Application architecture

- The UI infrastructure to build applications
- Higher level components such as
 - Wizards, User Settings, Data Access framework
- Windowing System
- And much more...

Flexible deployment and distribution

- Big releases can be deployed as usual
- Parts of the application (a plugin) can be updated and distributed individually
- New features can be released in between big releases
- Much more agile approach to releasing software

Available Frameworks

- Swing Application Framework (JSR-296)
- Spring Rich Client
- JGoodies Swing Suite
- Eclipse RCP
- Netbeans RCP

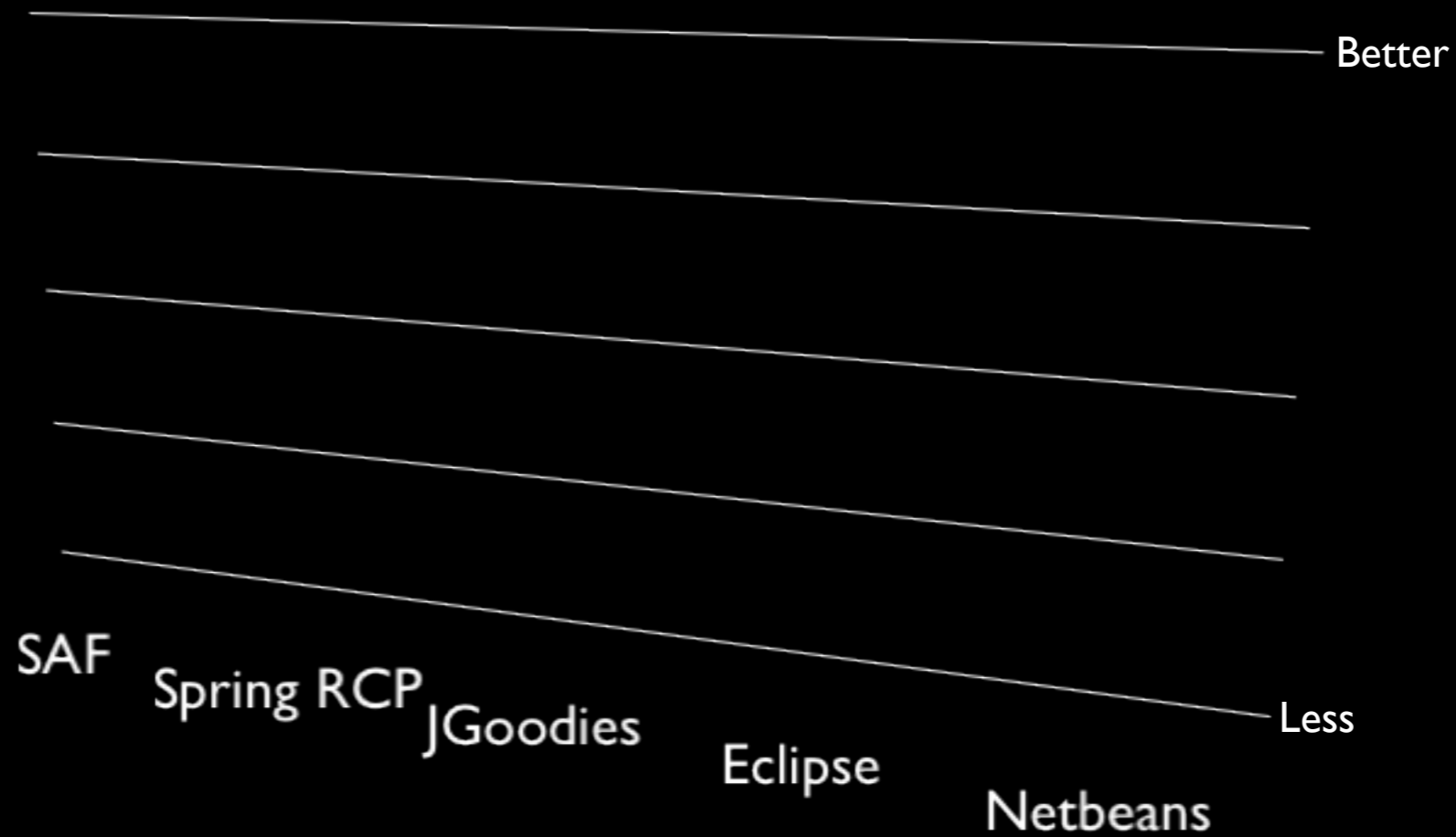
Available Frameworks

Better

Less

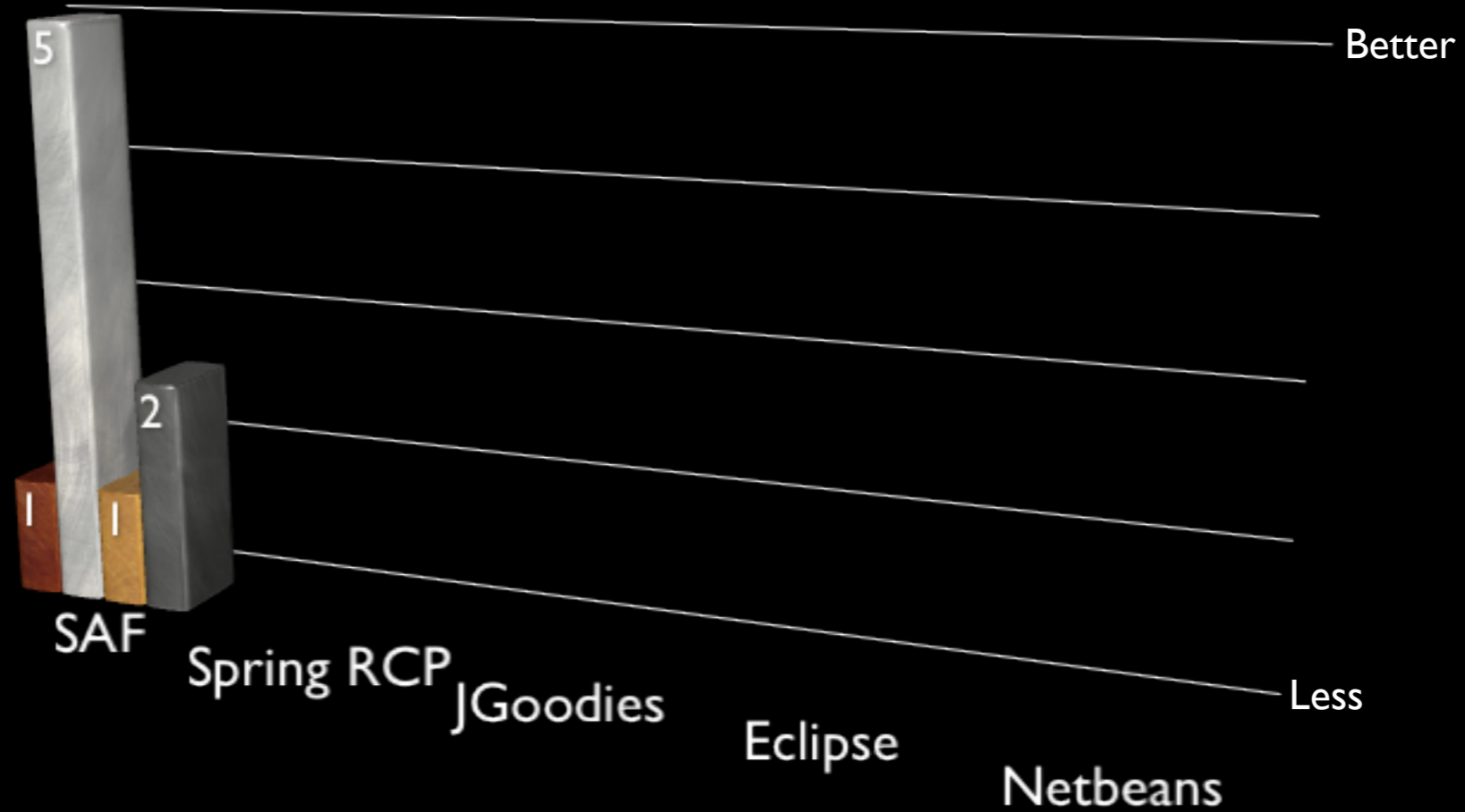
-  Features
-  Learning Curve
-  Maturity
-  Community

Available Frameworks



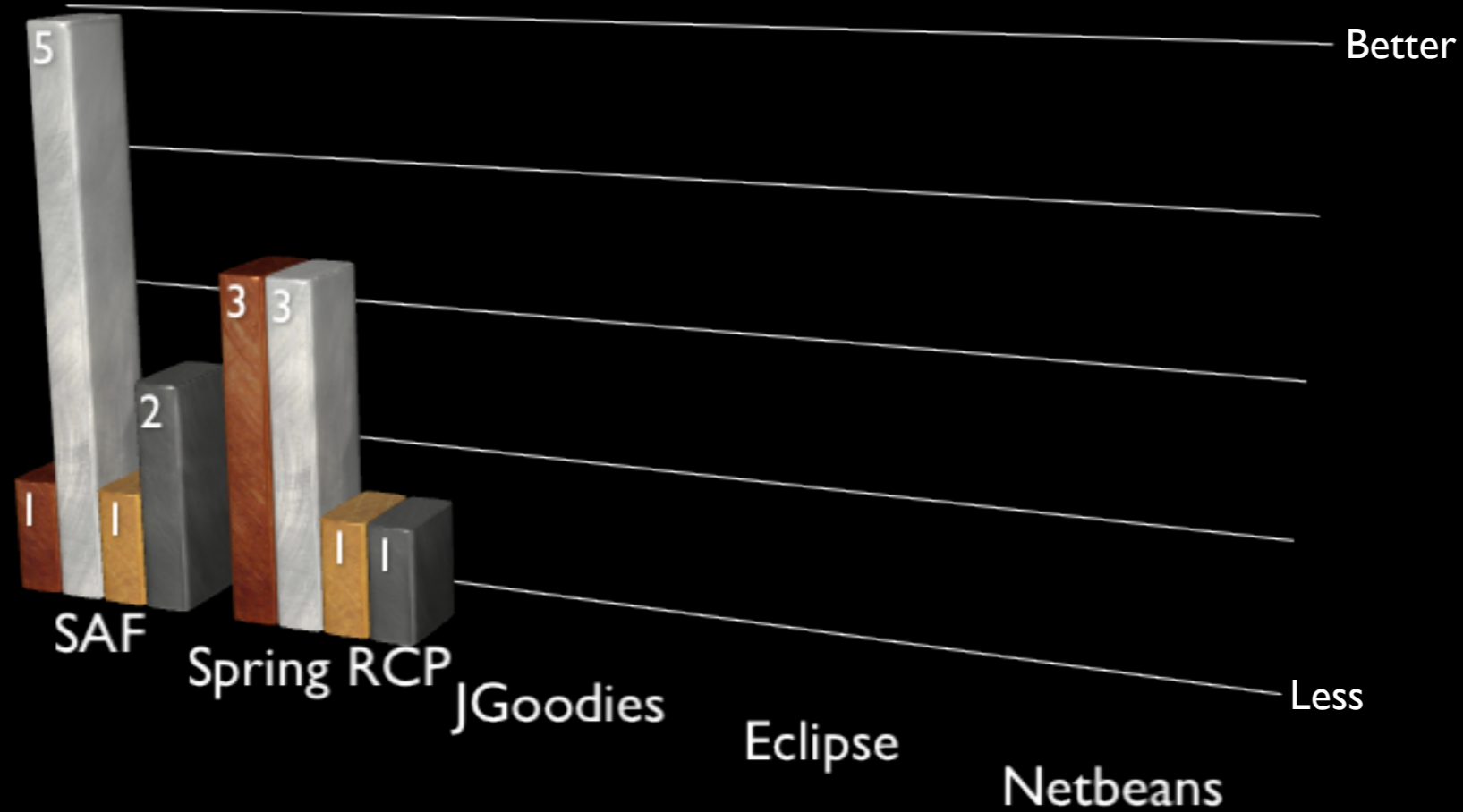
- Features
- Learning Curve
- Maturity
- Community

Available Frameworks



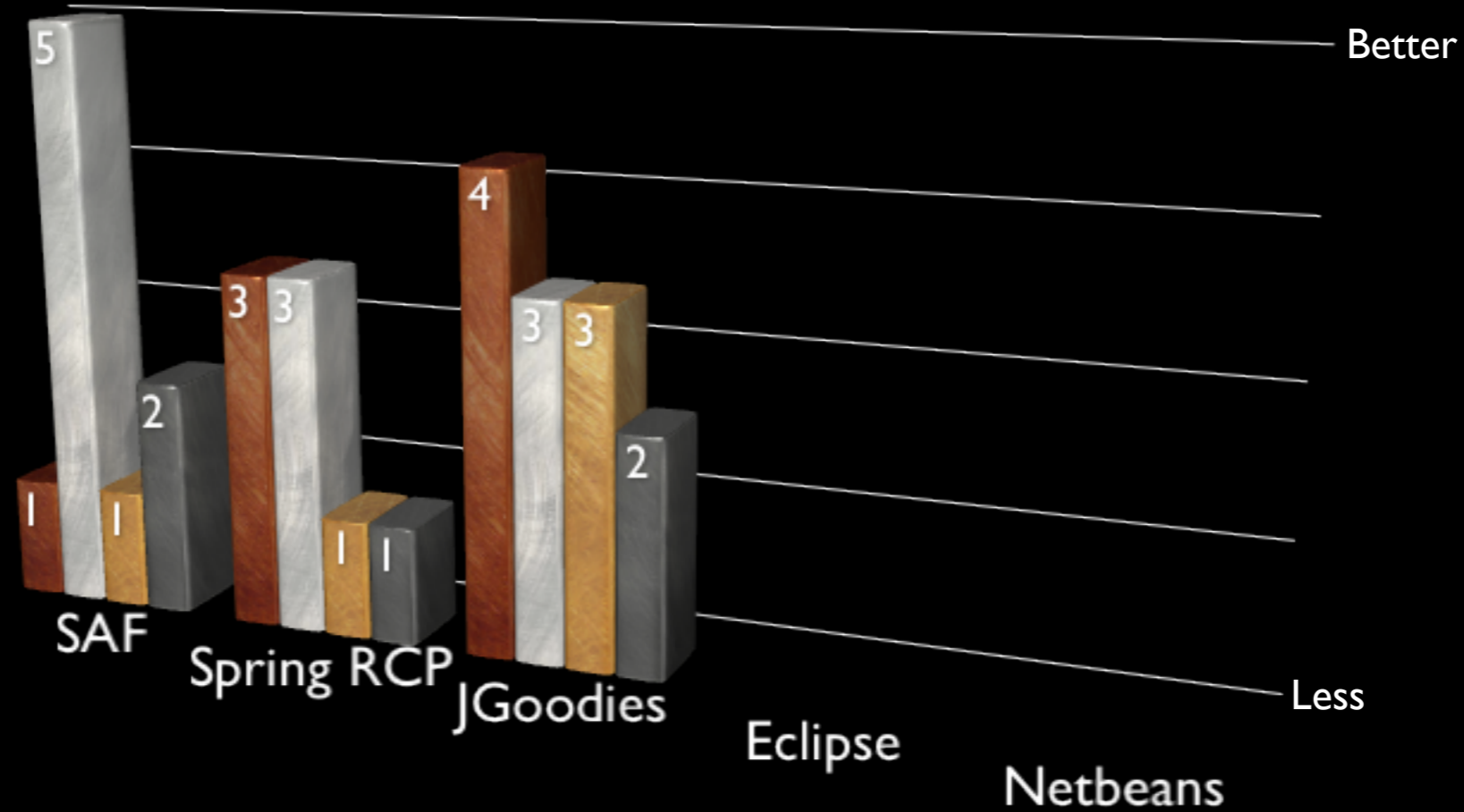
- Features
- Learning Curve
- Maturity
- Community

Available Frameworks



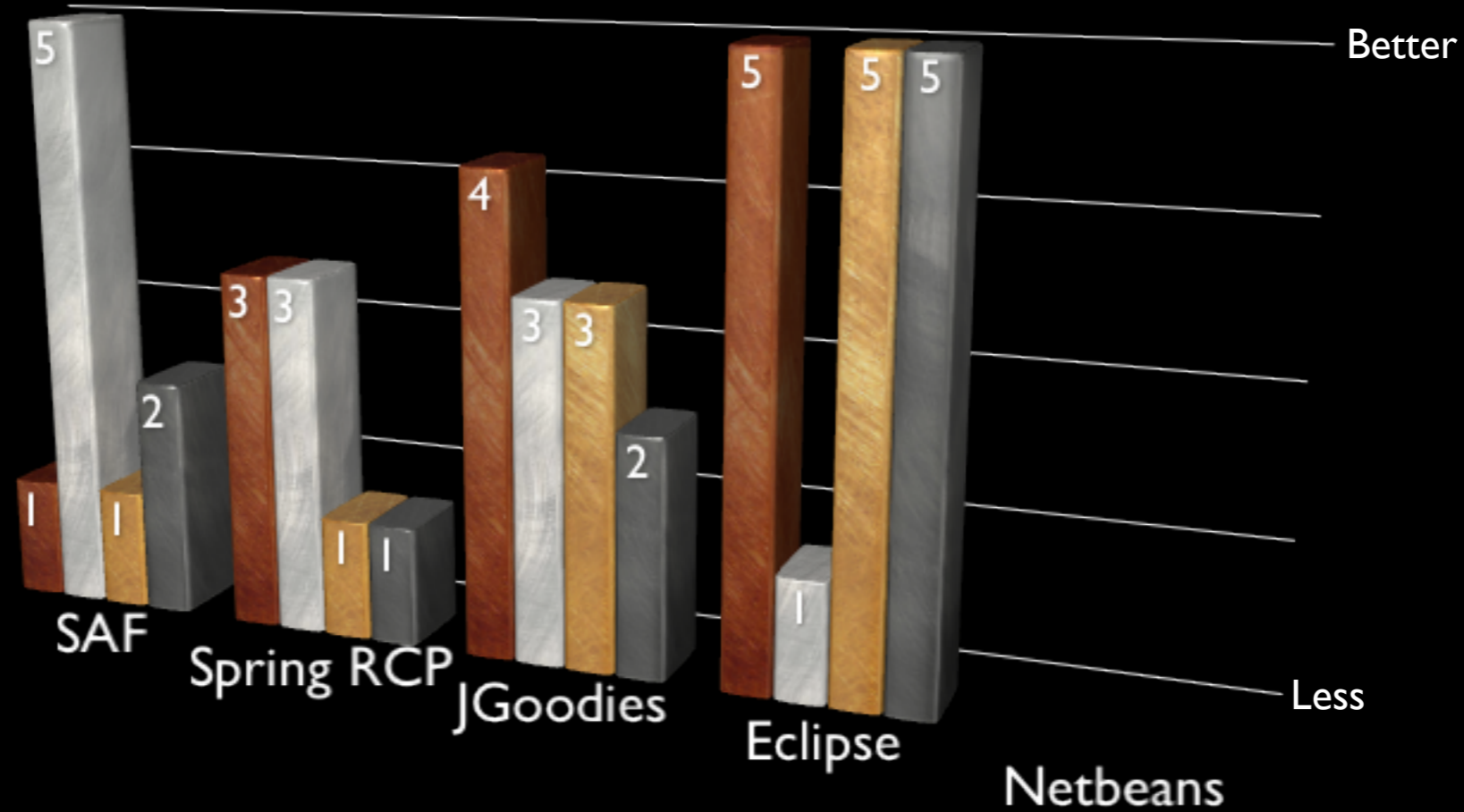
- Features
- Learning Curve
- Maturity
- Community

Available Frameworks



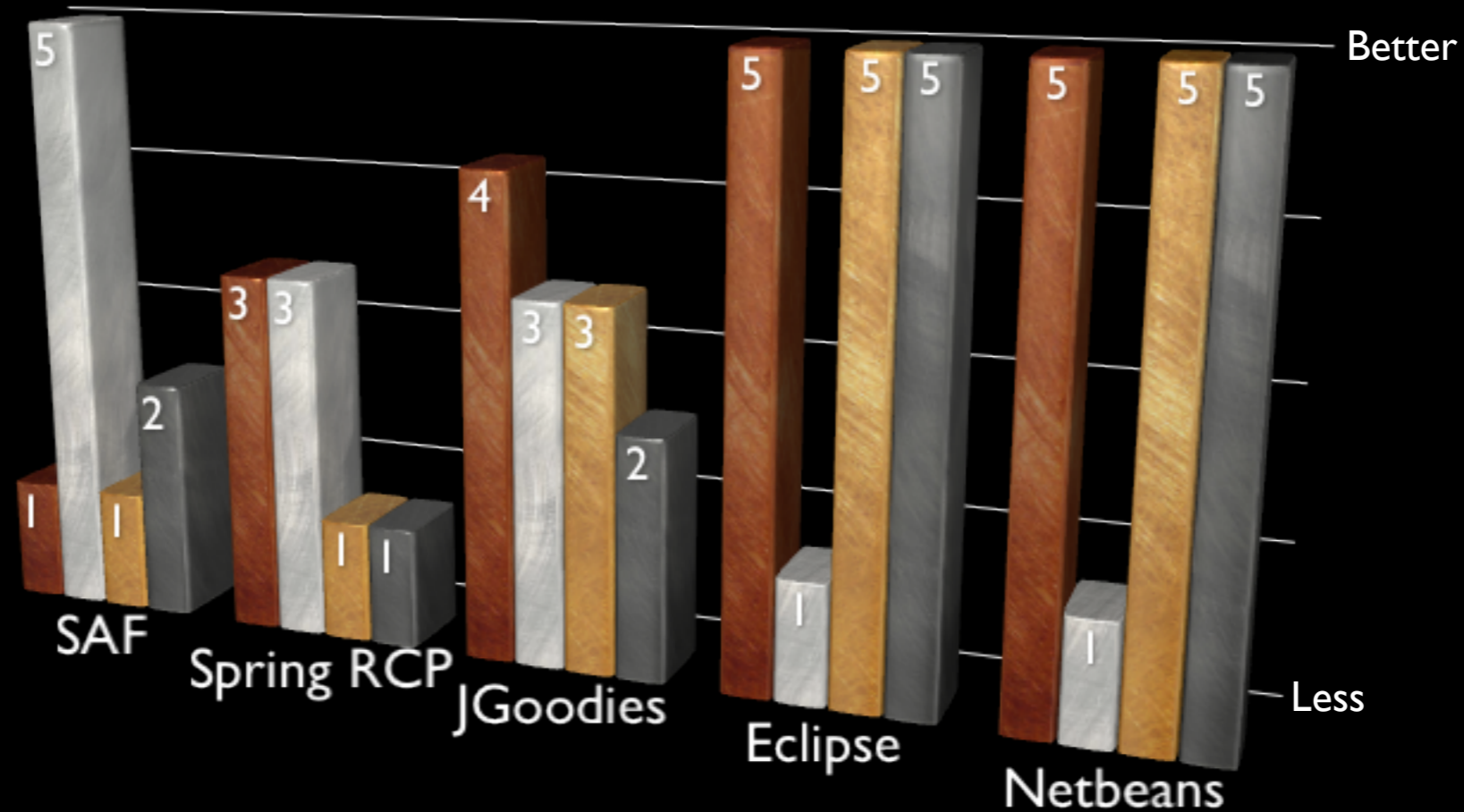
- Features
- Learning Curve
- Maturity
- Community

Available Frameworks



- Features
- Learning Curve
- Maturity
- Community

Available Frameworks



- Features
- Learning Curve
- Maturity
- Community

Eclipse RCP

- OSGi based container (Equinox)
 - Manage dependencies and lifecycle
 - Explicitly supports dynamic scenarios
- Highly adopted standard
- OSGi Bundle is the unit of modularization

Eclipse RCP

- Based on SWT, native look-and-feel
- Swing component can be embedded
- Requires Eclipse IDE to build (*)
- It is possible to run two or more versions of the same module in one application

Netbeans RCP

- Proprietary runtime based on Java Extension Mechanism
 - Manage dependencies and lifecycle
- NetBeans module is the unit of modularization
 - Soon to support OSGi bundles

Netbeans RCP

- Built using Ant or Maven, no IDE lockin (*)
- Java Webstart deployment out of the box
- It is possible to run two or more versions of the same module in one application
- Based on Swing

But Swing is slow...

The performance of Swing is
(for years) not an issue!

“the only problem with Swing is that there are a limited number of higher-level abstractions available that assist in making the toolkit simpler and easier to use” - The Spring Rich Client Team

Licensing

- Eclipse RCP
 - Eclipse Public License (EPL)
- Netbeans RCP
 - Common Development and Distribution License (CDDL)
 - GPLv2 with Classpath Exception

DEMO

- Creating a Netbeans RCP based “Hello World” application using Maven
- No dependency to NetBeans IDE
- Start with a empty directory
- Not pre-cooked in any way (*)

DEMO

(this page is for reference purpose only)

- `mvn -DarchetypeGroupId=org.codehaus.mojo.archetypes
-DarchetypeArtifactId=netbeans-platform-app-archetype
-DarchetypeVersion=1.2 -DarchetypeRepository=http://
repo1.maven.org/maven2 archetype:generate`
- `cd <directory>`
- `mvn install`
- `cd application`
- `mvn nbm:run-platform`

Real world experience using Netbeans RCP

- Requires initial investment
- Great community, books, tutorials
- Modularity is complex
- Unit Testing without the runtime
- The usual suspects will haunt you
 - Java Help, Java WebStart, Maven

Real world experience using Netbeans RCP

- We estimated the cost not to use an RCP to \$30K
- The UI quality would have been significant less

Conclusion

- For small applications the initial investment could be too high
- What is most important for you?
 - Can you reuse plugins from the community
 - SWT or Swing
 - OSGi or not
 - Build using Eclipse or Maven / Ant

Conclusion

There is no need to
reinvent the wheel
by writing UI
infrastructure code

Conclusion

- or -

Whatever you choose, it
is much, much better than
what you could achieve
yourself

Q & A