

inside eclipse

Erich Gamma

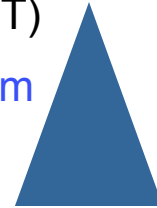
Eclipse Project Management Committee Member

IBM Distinguished Engineer

IBM Rational Software

what is eclipse?

- an IDE and more...
 - it's a [Java development environment](#) (JDT)
 - it's a general [tools](#) and integration [platform](#)
 - it's a general [application platform](#) (RCP)
- an [open source](#) community
- an [ecosystem](#) to enable a total solution
 - including products by some major tool vendors
- a [foundation](#) to advance the [eclipse platform](#)



the full eclipse spectrum – eclipse has grown...

- eclipse project
 - Platform
 - JDT
 - PDE
- eclipse tools project
 - VE
 - UML2
 - Hyades
 - CDT
 - GEF
 - EMF
- eclipse web tools project
 - BIRT (Business Intelligence Report Tools)
 - TPTP (Test Performance Tools Platform)
 - STP (SOA Tools Platform)
- eclipse technology project
 - Mylar
 - AJDT
 - AspectJ
 - Equinox
 - Pollinate (proposal)

⇒ **callisto** simultaneous release - ten major eclipse projects release at the same time at the end of June 2006

outline

- more than a Java IDE
- more than a tools platform (RCP)

eclipse perception changes



3.2 “eclipse is an even better application and tools platform and Java IDE”

3.0/3.1 “eclipse is a general application platform”

2.0 “eclipse is a general tooling platform”

1.0 “eclipse is a Java IDE”

broader application improves the platform

built to last

- deliver on time, every time
 - decisions in this release impact what we can do next release
 - must preserve architectural integrity

how buildings last

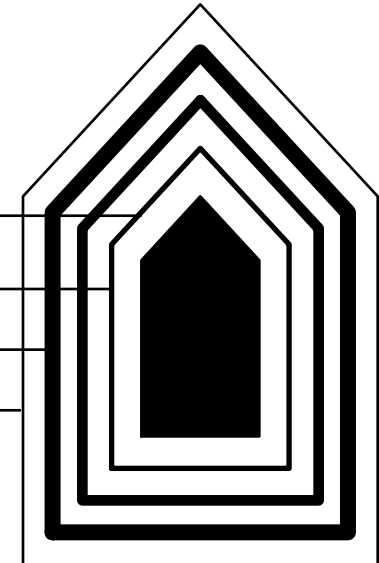
- **Stewart Brand: how buildings learn**
– what happens after they're built

stuff: furniture

services: electrical, plumbing (7-15y)

structure: foundation, load bearing walls (30-300y)

site: geographical setting (forever)



Site

- **layers:**

- evolve at different rates during the life of a building
- shear against each other as they change at different rates
- an adaptive building must allow **slippage**
- a building that lasts is adaptive and can change over time
- lasts for generations without total rebuilding

example

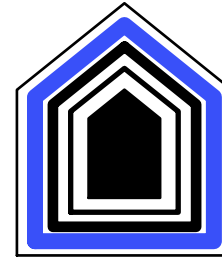


built 1860



today

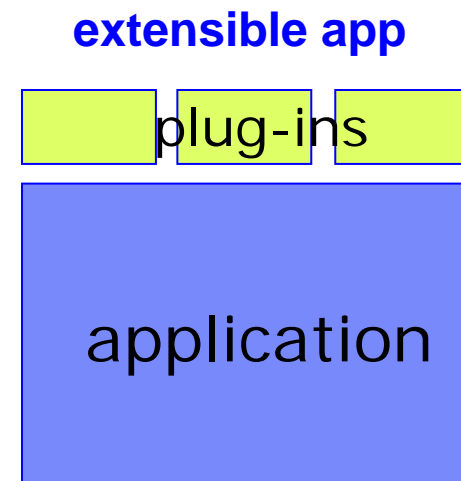
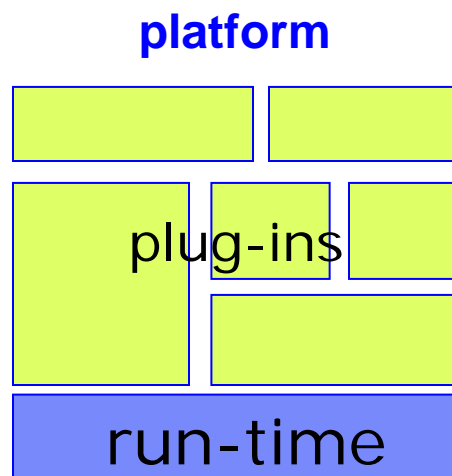
structure foundation




- the eclipse plug-in architecture
- everything is a plug-in
 - simple and consistent

platform vs. extensible application

- eclipse rich client platform
 - it has an open, extensible architecture
 - built out of layers of plug-ins



everything is a plug-in

- plug-in 
 - set of contributions
 - smallest unit of eclipse functionality
 - declares its pre-requisites

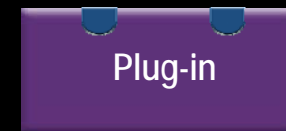
Plug-in

Plug-in

Plug-in

everything is a plug-In

- extension point 
 - named entity for collecting contributions



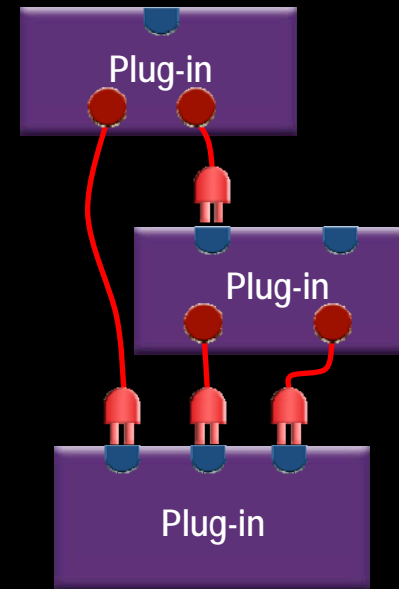
everything is a plug-in

- extension ●
 - a contribution



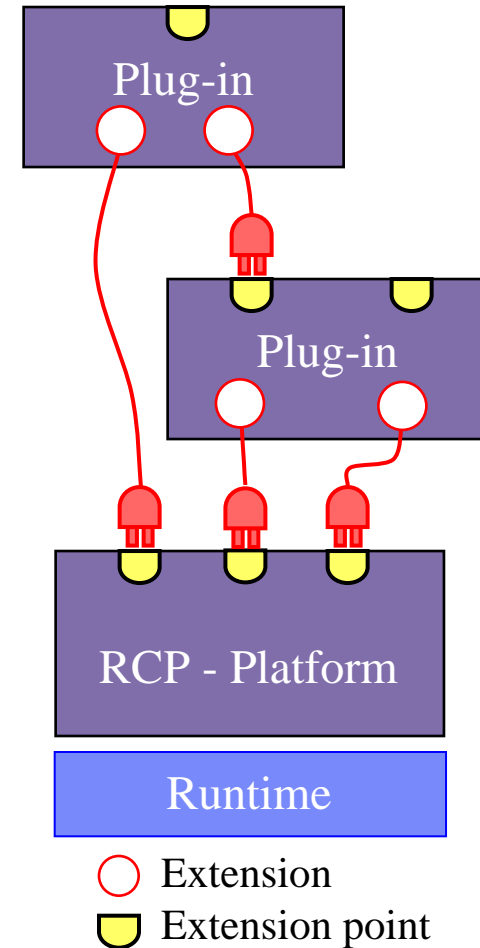
everything is a plug-in

- the platform run-time is making the connection



eclipse plug-in architecture

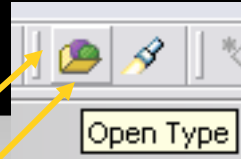
- **plug-in == component**
 - set of contributions
 - smallest unit of Eclipse function
 - details spelled out in plug-in manifest
 - big example: mail client
 - small example: action to calculate the number of lines of a mail
- **extension point** – named entity for collecting contributions
 - Example: extension point to add additional spam filtering tools
- **extension** – a contribution
 - Example: a specific spam filter tool
- **RCP - Platform** – set of standard plug-ins
- **Runtime** – controls and manages contributions



scalability

user visible
appearance

```
<action  
  toolbarPath="search"  
  icon="icons/opentype.gif"  
  tooltip="Open Type"  
  class="org.eclipse.jdt.OpenTypeAction"/>
```



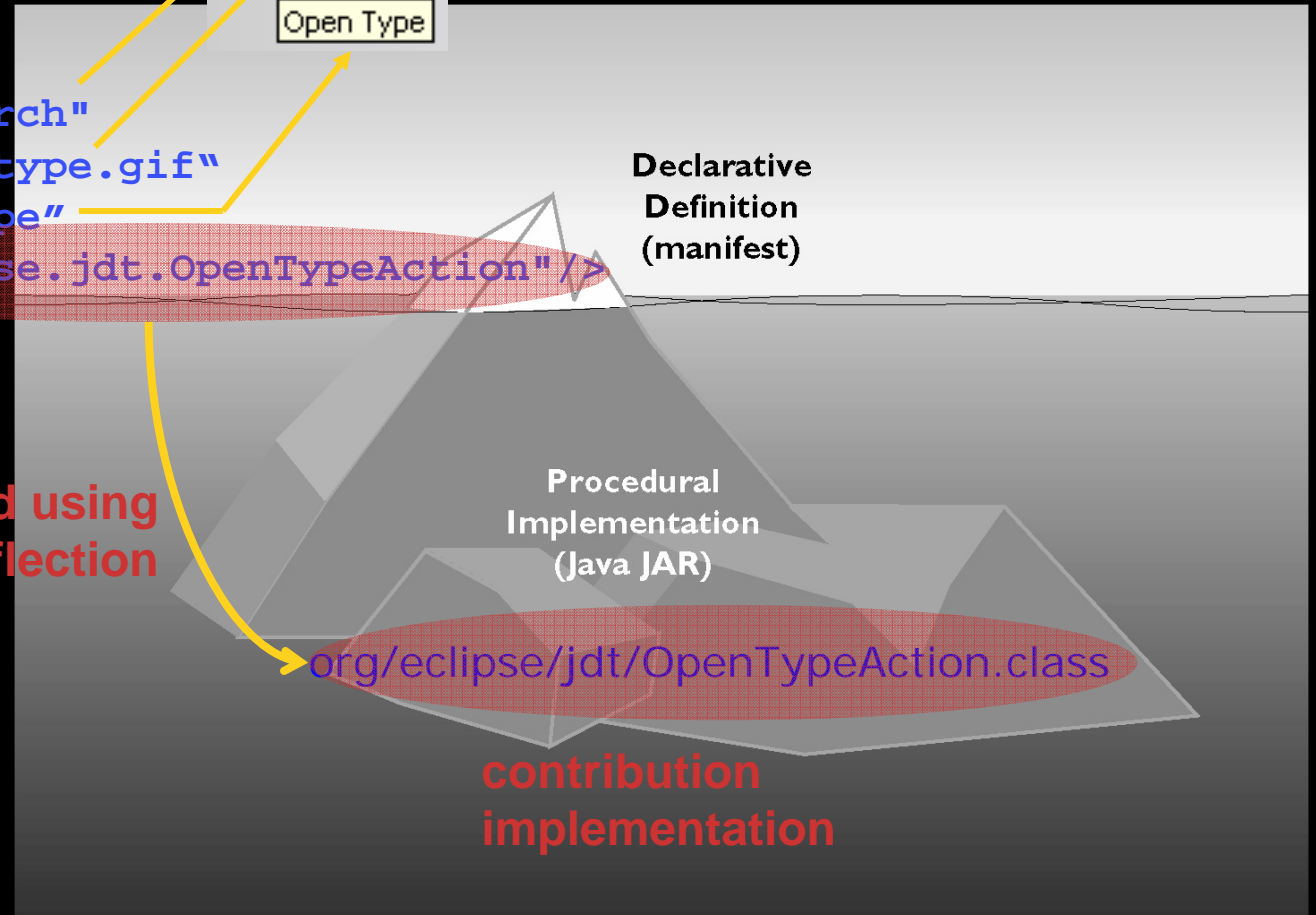
Declarative
Definition
(manifest)

lazily instantiated using
reflection

Procedural
Implementation
(Java JAR)

org/eclipse/jdt/OpenTypeAction.class

contribution
implementation



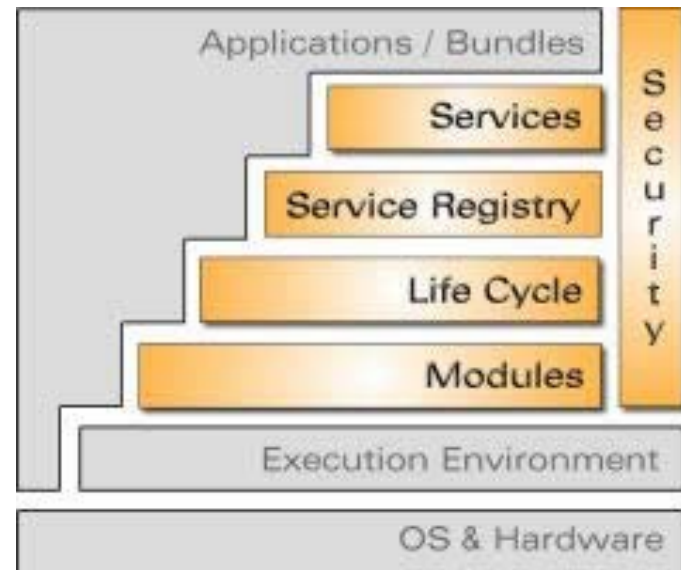
Runtime Component Model

OSGi – Open System Gateway Initiative

⇒ Brings modularity to Java

- Named, versioned bundles
 - Dependency management
 - Explicit imports/exports
 - Built-in security
 - Dynamic
- Independent industry standard
 - Has become popular in cell phone system management and other areas

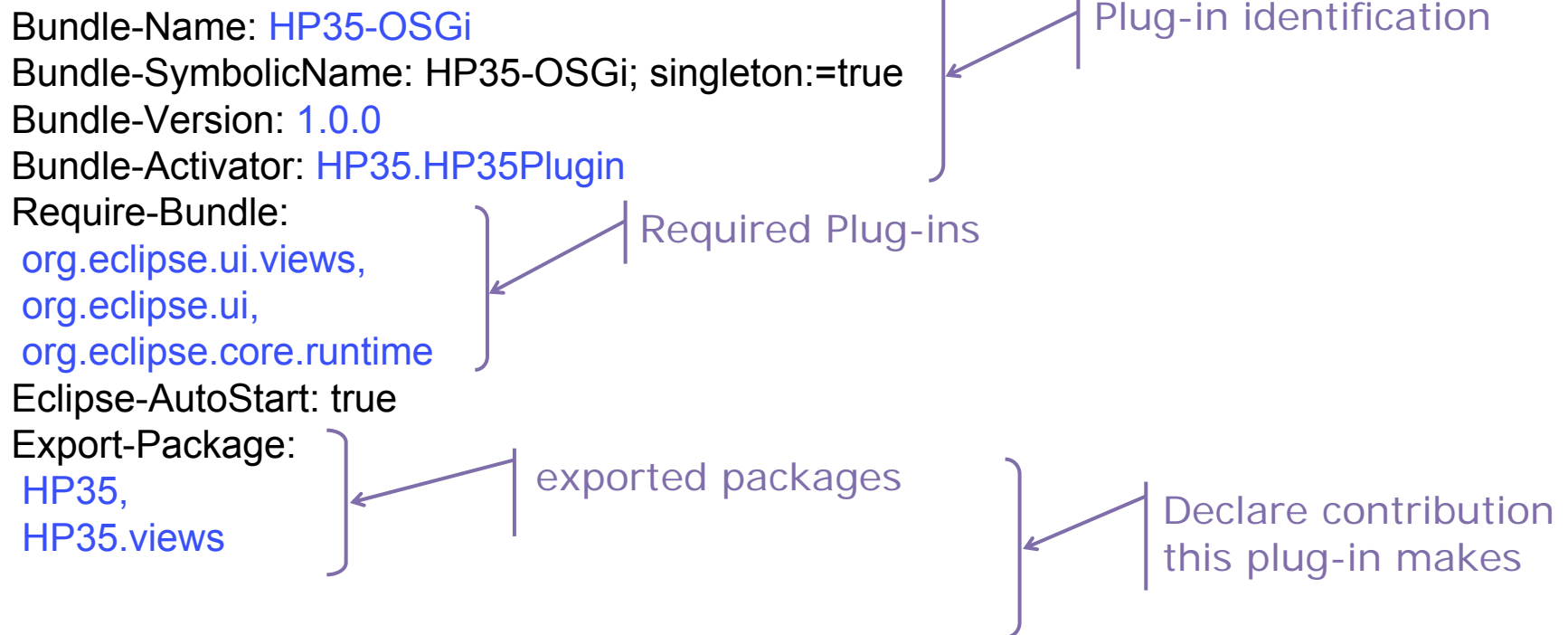
⇒ These are exactly the characteristics we want for advanced Smart Clients



eclipse plug-in architecture: describing a plug-in

- **3.1 separates plug-in description into two manifests**
 - prior to 3.1 a single manifest was used
- **plug-in manifest files**
 - to describe *component attributes*
 - OSGI META-INF/MANIFEST.MF
 - to describe *extensions*
 - plugin.xml
- **Plug-in development tools**
 - provide a combined editor
 - perform semantic checking of the contents

manifests – component model - OSGi



manifests – extensions \Rightarrow plugin.xml

```
<plugin>
```

```
  <extension>
```

```
    <view
```

```
      name="HP35"
```

```
      icon="icons/calculator.jpg"
```

```
      category="HP35"
```

```
      class="HP35.views.HP35View"
```

```
      id="HP35.views.HP35View">
```

```
    </view>
```

```
  </extension>
```

Declare contribution
this plug-in makes



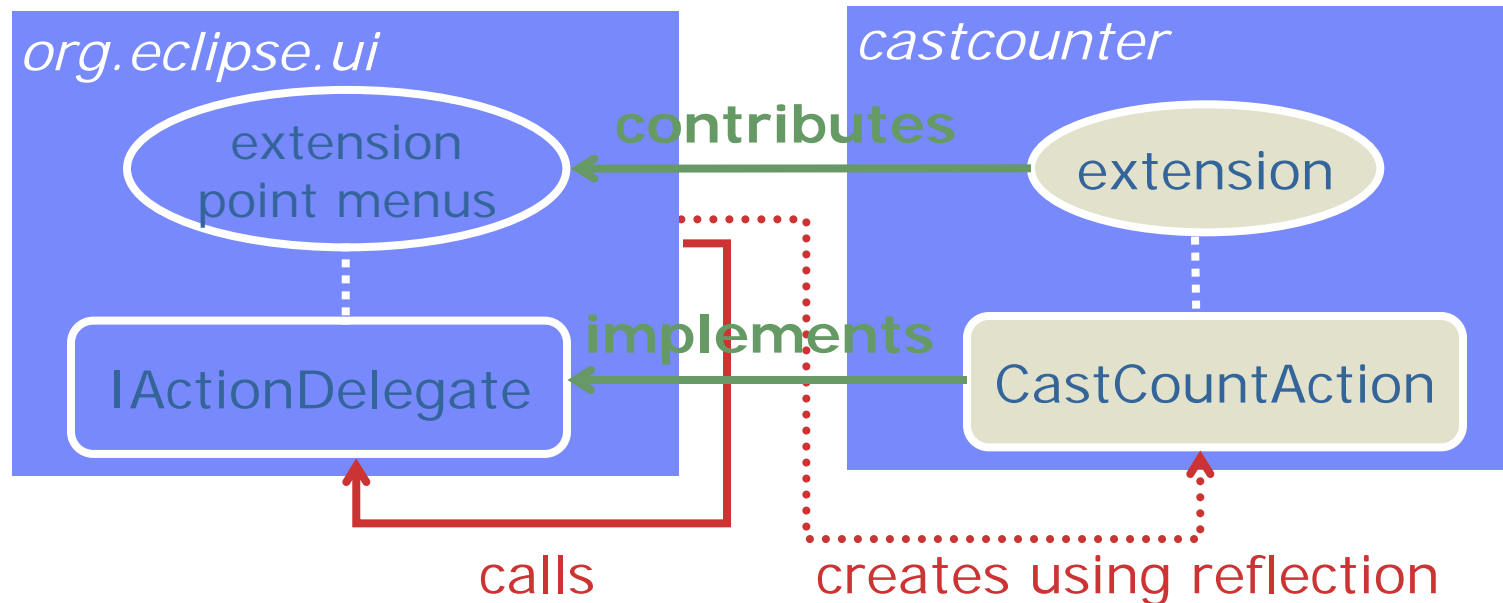
```
  <extension-point id="operators" name="Operators"  
    schema="schema/operators.exsd"/>
```

Define new extension point
open for contributions



```
</plugin>
```

Contributing an extension



- plug-in **org.eclipse.ui**
 - declares extension point **org.eclipse.ui.popupmenus**
 - declares interface **IActionDelegate**
- plug-in **castcounter**
 - implements interface **IActionDelegate**
 - contributes class **CastCountAction** to extension point
- plug-in *org.eclipse.ui* instantiates **CastCountAction**

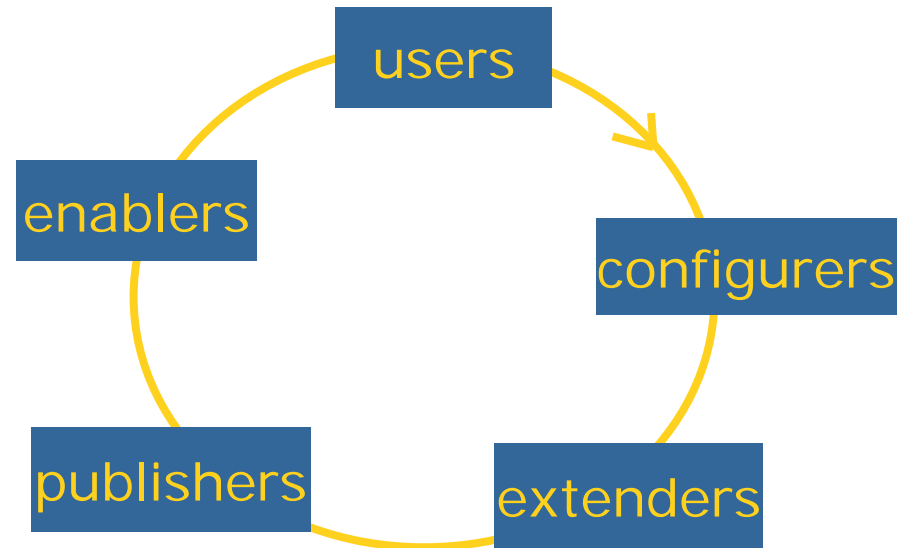
developing and deploying plug-ins

- ***plug-in development environment*** (PDE)
 - specialized tools for developing eclipse plug-ins
 - e.g. editor for plug-in manifest files
 - templates for new plug-ins
 - implemented as plug-ins atop eclipse platform and JDT
 - PDE runs and debugs another eclipse application

- ***features*** group plug-ins into installable chunks
 - features downloadable from url addressable location
 - obtain and install new or update existing plug-ins

contribution cycle...

- the *contribution cycle*
- **publish** the plug-in
 - create a feature
 - groups plug-ins into installable chunks
 - create an update site
 - contains zips for features and plug-ins
- **enable** extensions
 - define extension points
- extenders can extend your extensions



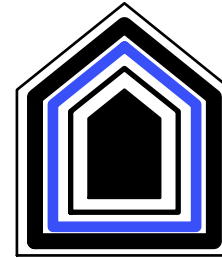
managing plug-ins: install/update

- **features** group plug-ins into installable chunks
 - feature manifest file
- plug-ins and features bear version identifiers
 - major . minor . service
 - multiple versions may co-exist on disk
- features downloadable from URL addressable location
 - using Eclipse Platform update manager
 - obtain and install new plug-ins
 - obtain and install patches & updates to existing plug-ins
- support for update site mirroring & shared installations

ready to use; easy to adopt

- component model
 - simple but consistent
- plug-ins have to be easy to develop \Rightarrow tool support
 - Java Development Tools (JDT) + Plug-in development environment (PDE)
- scale-up to thousands of installed plug-ins
 - the problem is start-up time
 - lazy loading
- plug-ins have to be easy to manage and discover
 - install, update, disable
 - built-in install/update support

services plumbing: APIs



- APIs matter
 - define consistent, concise API
 - define API conventions (*.internal.* in eclipse)
 - don't expose the implementation
 - develop implementation and client at the same time
- define APIs for stability
 - binary compatibility is highest priority
 - we would rather provide less API than desired (and augment) than provide the wrong (or unnecessary) API and need to support it indefinitely

component centric

- component centric development
 - a team is responsible for one or more component
- dependencies through APIs
 - ensures high velocity development inside a component
 - *eclipse 3.1 provides tools support to check for API access violations*
 - *as you type*
- define producer/consumer relationships among components
 - tension among components is healthy for coming up with good component interfaces/APIs

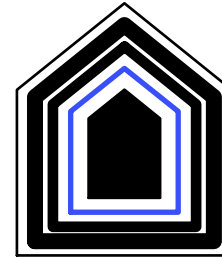
APIs first

- APIs don't just happen; we need to design them
- specifications with precisely defined behavior
 - what you can assume (and what you cannot)
 - it works \neq API compliant
 - documented classes \neq API
 - "provisional API" = API that didn't make it for a release
- must have at least one client involved, preferably more
- need an API advocate
 - we all care about having sustainable APIs
 - need someone who lives and breathes APIs

example: API evolution in the Java Development Tools

- new APIs
 - AST (Abstract Syntax Tree)
 - AST rewriting
 - code manipulation
- open-up
 - contribute to quick fix/quick assist
 - contribute to code assist
- push-down
 - make JDT specific support available to other languages:
 - template processors
 - linked editing

stuff, furniture - UI



- eclipse extension architecture is contribution based
 - extensions contribute to the workbench
 - the workbench manages and presents the contributions
- enables UI evolution
 - 3.0 new look
 - 3.1 new preferences

keeping the house clean...

- blur the boundary between the Platform and contributions
 - plug-ins should fit in (and together) naturally
 - this is a shared responsibility for all plug-in developers

housekeeping rules

- **sharing rule:** Add, don't replace
- **invitation Rule:** Whenever possible, let others contribute to your contributions
- **fair Play Rule:** All clients play by the same rules, even me.
- **Relevance Rule:** Contribute only when you can successfully operate
- **Integration Rule:** Integrate and don't separate

outline

- more than a Java IDE
- more than a tools platform (RCP)

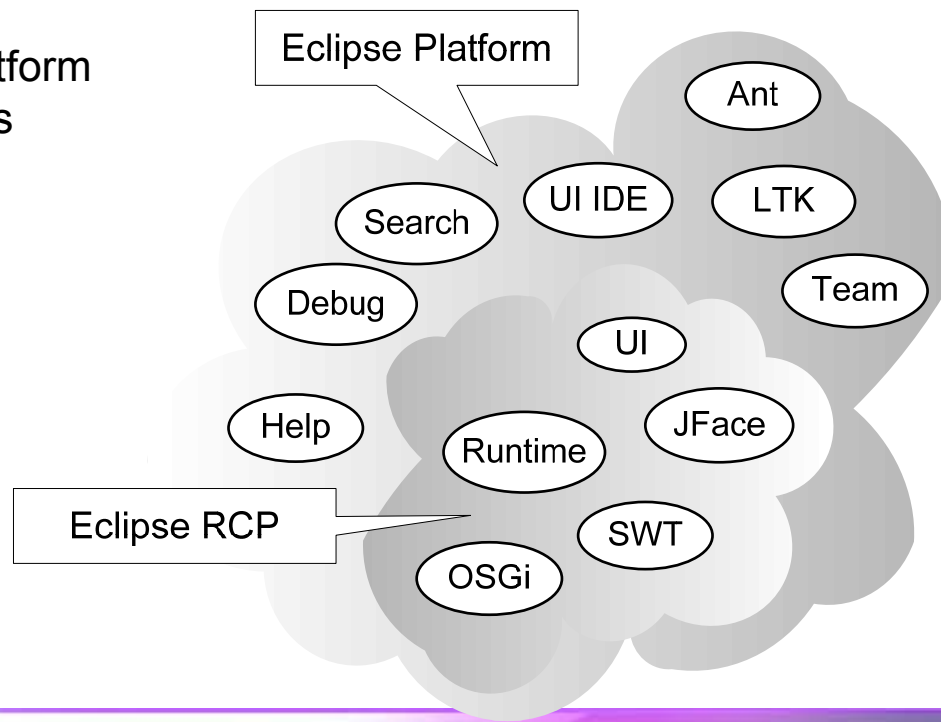
a brief history of RCP

- Eclipse 2.1 (2003): Eclipse is an IDE platform. Hackers (aka smart people) in the eclipse community started building non-IDE apps based on Eclipse IDE technology.
- Eclipse 3.0 (2004): We made this real by:
 - factoring out the IDE-specific aspects from the workbench
 - you get a blank slate: an empty workbench
 - but with a very rich set of features to fill it with
 - RCP-specific APIs for configuration and lifecycle notification
 - integrating with the new OSGi runtime
 - cleaning up dependencies in other components (e.g. Help and Update)
- Eclipse 3.1 (2005): improved tooling, some new APIs

what is RCP?

while the Eclipse Platform is designed to serve as an open tools platform, it is architected so that its components can be used to build just about any client application. The minimal set of plug-ins needed to build a rich client application is collectively known as the **Rich Client Platform**.

- a subset of the full Platform
- there are other subsets



characteristics of a smart client application

- desktop app (not in web browser)
- run on multiple platforms, devices, and configurations
- rich UI with consistent metaphors
- tight integration with desktop OS (e.g. DnD, System Tray)
- easy deployment and server-centric management -> lower TCO
- makes use of local filesystem and other devices (printer, card reader)
- makes use of server-side resources and data too
- reduced round trips to the server
- responsive UI
- occasionally disconnected operation
 - securely caching data, knowing when to invalidate
 - indicating potentially stale data to user

Example RCP apps

IBM Workplace Client Technology, Client Administrator

The screenshot displays the IBM Workplace Client Administrator interface. At the top, there are navigation tabs for 'OpenFinancial Teller', 'S1 Enterprise Teller', 'Customer Service', and 'My Workplace'. The main header features the IBM logo and the text 'open financial network'. The interface is divided into several panes:

- Customer Search:** Includes input fields for 'Enter Lastname:', 'Enter Firstname:', 'Enter SSN:', and 'Enter Account:', each with a 'Search' button. Below these is a section for 'All Accounts:' with a 'Search' button and a message '10 accounts are found'.
- Customer Profile:** Displays information for 'John William' with SSN '123-45-6789'. It includes a profile picture, contact details (Address: 345 Anderson St, New York, NY-10001, USA; Email: abc@xyz.com; Phone Home: 123-456-7890; Phone Work: 231-456-7890), and a list of accounts.
- Account Details:** Shows details for account number '294311310736', which is a 'Checking' account with a balance of '6400.0' and an available balance of '5400.0'. It also lists 'Recent Transactions' with columns for Date, Transaction Type, Amount, and Balance.
- Customer Search Results:** A table listing customer names and SSNs.
- Account Search Results:** A table listing account numbers, types, and balances.
- Deposit/Withdraw:** A section for performing transactions, showing account details and a 'Deposit' button.

At the bottom of the interface, there is a taskbar with icons for 'Extras', 'Shutdown', 'Logoff', 'Lockup', 'Task List', 'Change Password', 'Locale', and 'Keyboard'.

- Dynamic, user-based provisioning
- Locked-down desktop
- Strong security requirements
- Restricted customizability

Maestro – NASA Space Mission Management

- Responsive UI
- Open Platform for space mission monitoring
- Bringing together different teams in NASA/JPL

The screenshot displays the Maestro software interface. The main window shows a grayscale image of the lunar surface with a network of yellow lines connecting 34 numbered sites. A green line traces a path through these sites. A blue box highlights a specific area on the map. On the right, the 'EDR Search View' panel is active, showing a table of mission data. Below the table, there are several 'Image View' windows, one of which shows a close-up of the lunar surface with shadows.

Product ID	Instrument	Sol	Seq ...
2F134356767E7FF2600P1212L0M1	FRONT_HAZCAM_LEFT	90	p1212
2F134449644EFF2700P1212L0M1	FRONT_HAZCAM_LEFT	91	p1212
2F134614869EFF2700P1403L0M1	FRONT_HAZCAM_LEFT	93	p1403
2F134615161EDN2700P1131L0M1	FRONT_HAZCAM_LEFT	93	p1131
2F135147950EDN2700P1131L0M1	FRONT_HAZCAM_LEFT	99	p1131
2F135148174ESF2700P1127L0M1	FRONT_HAZCAM_LEFT	99	p1127
2F135149189EDN2700P1141L0M1	FRONT_HAZCAM_LEFT	99	p1141
2F135149794EDN2700P1141L0M1	FRONT_HAZCAM_LEFT	99	p1141
2F135150380EDN2700P1141L0M1	FRONT_HAZCAM_LEFT	99	p1141
2F135150997EDN2700P1141L0M1	FRONT_HAZCAM_LEFT	99	p1141
2F135151610EDN2700P1141L0M1	FRONT_HAZCAM_LEFT	99	p1141
2F135152416EDN2700P1141L0M1	FRONT_HAZCAM_LEFT	99	p1141
2F135152602EFF2700P1212L0M1	FRONT_HAZCAM_LEFT	99	p1212
2F135153765EDN2700P1111L0M1	FRONT_HAZCAM_LEFT	99	p1111

eclipse rich client platform

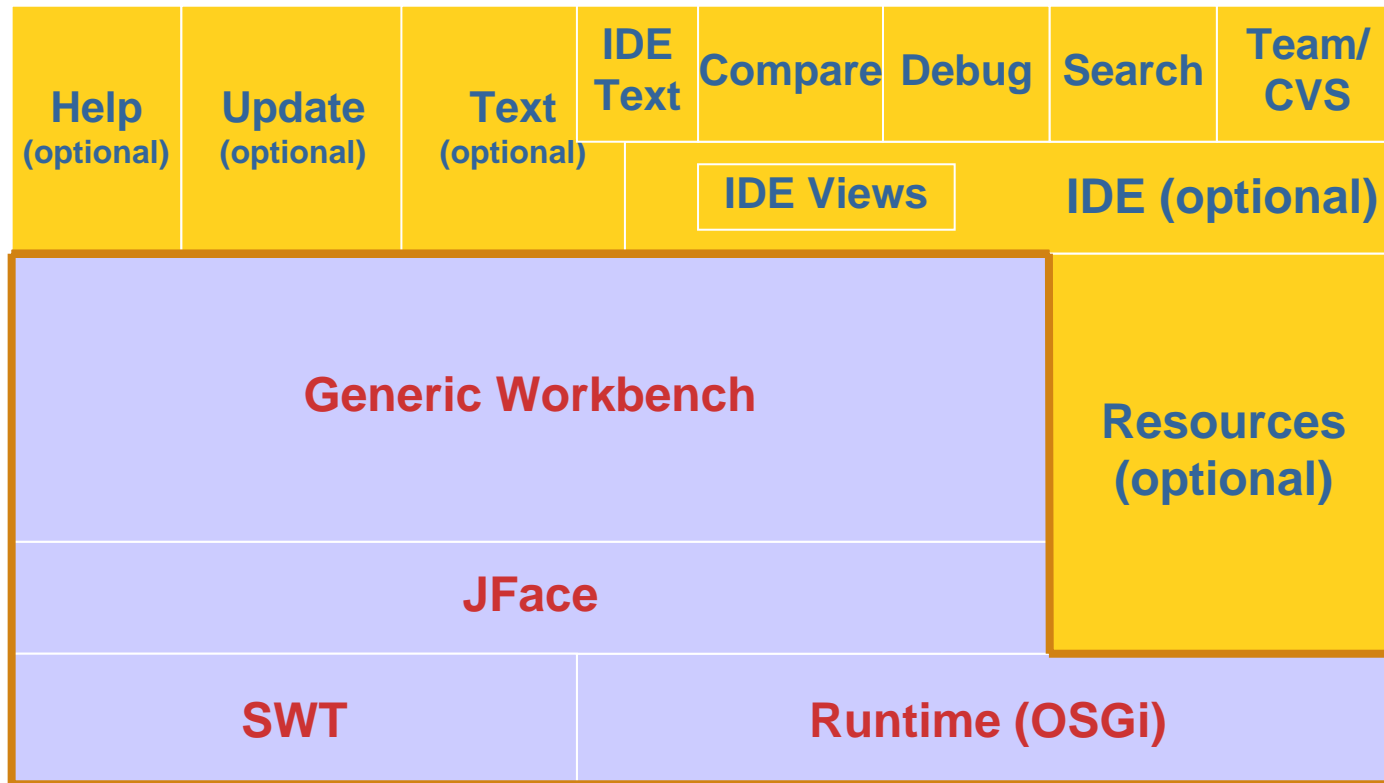
a rich client platform needs a strong component model with the following major characteristics:

- ✓ specified interfaces: a component must declare its public API and how it can be extended
- ✓ lazy loading: components are loaded on demand not on startup
- ✓ versioning: prerequisite components are reference by name and version
- ✓ dynamic detection: components are detected dynamically (no need to restart)

Additionally the following issues must be addressed:

- ✓ managing: install, update, remove & discover components
- ✓ development: IDE to develop components

RCP components + optional components



rcp in action: hp35



モデル35操作ガイド

電池が消耗すると、全ての小数点が点灯、不合理的な演算を行なうとゼロを表示し点滅、操作(リセット): **[OL]**

[CL] 表示をクリアー、
[CHS] 表示の符号を反転、負のデータを入力する場合には、最初に**[CHS]**キーを操作する。

[CLR] 全てのレジスタをクリアー、
[EEX] データxに10の指数を設定するとき、指数入力の前に操作、負の指数は指数入力の前にさらに**[CHS]**キーを操作。

T		-T	動作レジスタはX, Y, Z, Tの4個で、他に定数ストア用として
Z		-Z	Sレジスタ1個付き。
Y		-Y	
X		-X	x, y, z, t, sはそれぞれX, Y, Z, T, Sレジスタの内容、表示されるのは常にXレジスタの内容。
S		-S	

[ENTER] **[x↔y]** **[R↓]** **[STO]** **[RCL]**

[CL], **[STO]**, **[ENTER]**にすぐ続く場合を除き、Xへのデータ入力または**[RCL]**により各動作レジスタの内容は自動的に上にシフト。

[+] **[-]** **[×]** **[÷]** **[√]** **[ln, sin*など]** *三角関数キーの場合には、TレジスタにもZレジスタと同じ内容が入る。角度の単位は全て度(°)。

[f] **[s]** **[y]** **[x]** **[y+x]** **[y-x]** **[y×x]** **[y÷x]** **[x-1(x)]**

例題: $(2+3) \times 4/5 \times 4^{-1.5} = 1$

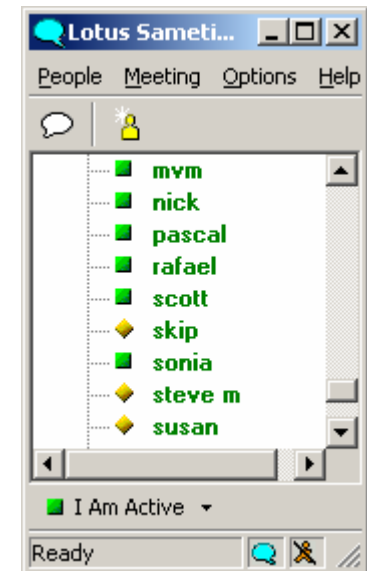
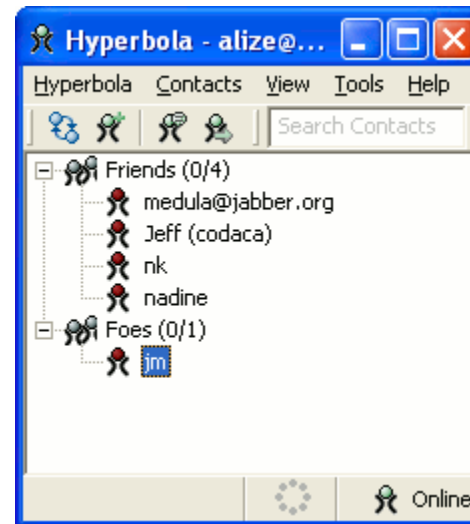
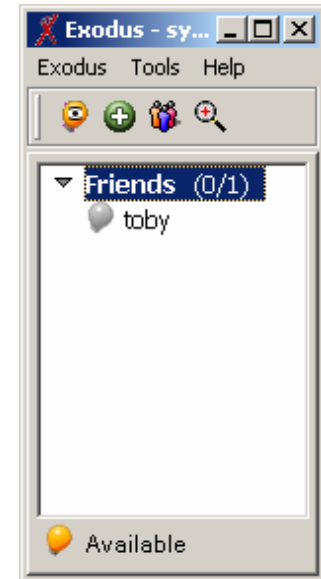
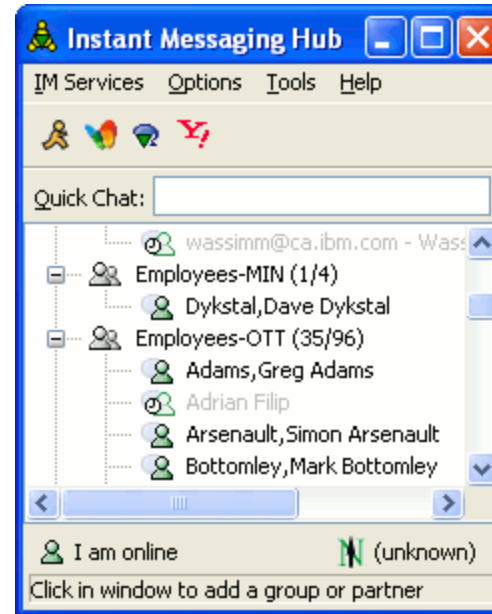
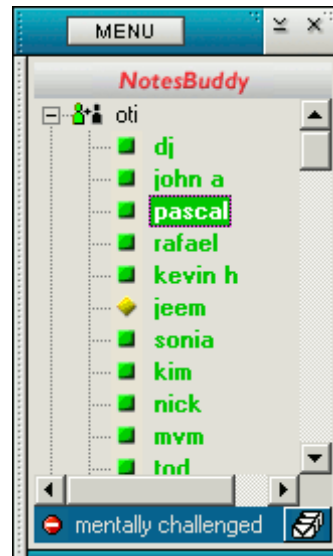
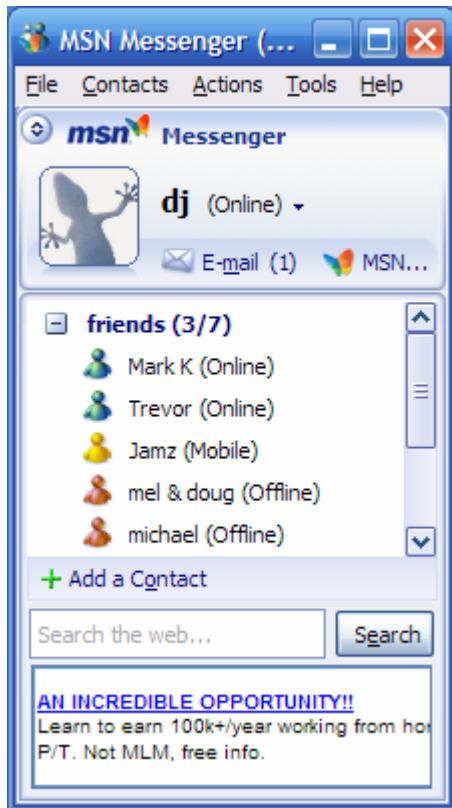
T																			
Z																			
Y		2	2	5	20	4	4		8	-1.5	-1.5	8							
X	2	2	3	5	4	20	5	4	30	.5	8	-8	-1.5	-1.5	4	.125	1		
操作	2	↑	3	+	4	×	5	÷	30	Sin	÷	CHS	1.5	↑	4	x ^y	X		

YOKOGAWA - HEWLETT - PACKARD
 3.75V 500MW
 MADE IN U.S.A. PATENT PENDING

UI Components

- SWT - **S**tandard **W**idget **T**oolkit
- JFace – Framework providing higher-level UI abstractions
- Workbench – Provides reusable and extensible UI metaphors
- Text - Framework(s) for building high-function text editors
- UI Forms - Framework for building forms-based views and editors
- GEF - Framework for building rich graphical editors

native integration
can you tell?

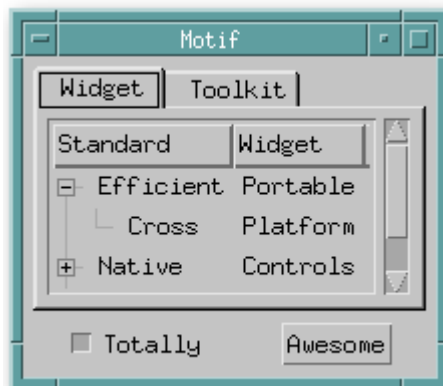
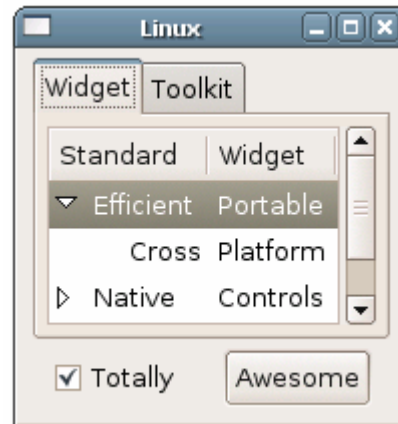


RCP means you can't tell

UI Components

SWT - **Standard Widget Toolkit**

⇒ **Platform-independent native widget toolkit**



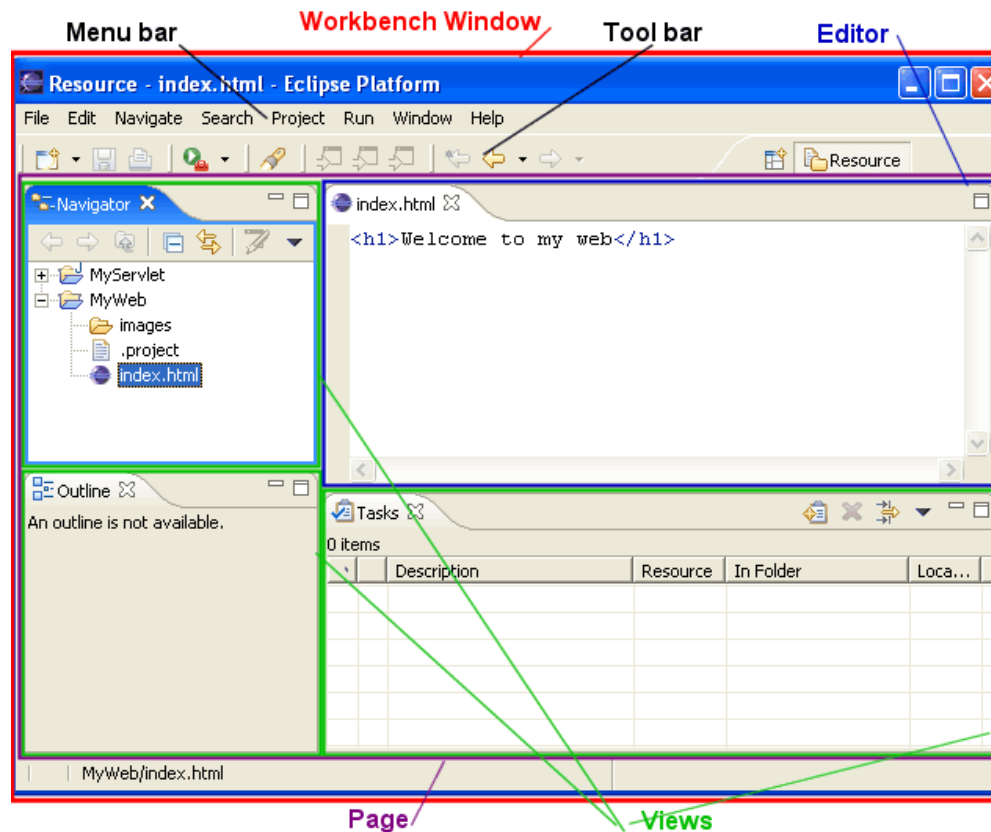
JFace

⇒ **Framework on top of SWT providing higher-level UI abstractions**

- Application window: menu bar, tool bar, content area & status line
- Viewers (MVC pattern)
- Actions, action bars (abstracts menu items, tool items)
- Preference and wizard framework

UI Components

Workbench



⇒ **Defines reusable and extensible UI metaphors**

Leverages extension point mechanism and JFace abstractions.

Provides:

- Views
- Editors
- Action sets
- Perspectives
- Wizards
- Preference pages
- Commands and Key Bindings
- Undo/Redo support
- Presentations and Themes
- Activities (aka Capabilities)
- ...

Is **Dynamic-aware**: responds to registry changes and adds/removes views, action sets, etc. accordingly

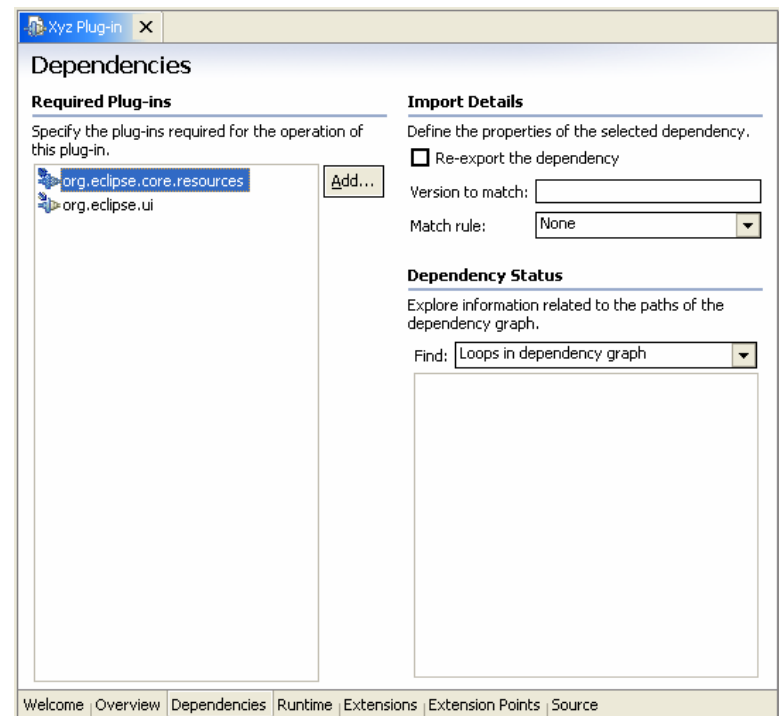
UI Components

UI Forms

⇒ **Framework for building forms-based views and editors**

- Form consisting of multiple FormParts
- Extra widgets:
 - FormText (marked-up text)
 - ScrolledForm
 - Section
 - MasterDetailsBlock
- Extra layouts:
 - TableWrapLayout (HTML-like)
 - ColumnLayout (newspaper-like)
- Flat look, lightweight borders
- Forms-based multi-page editor

- Used extensively in PDE



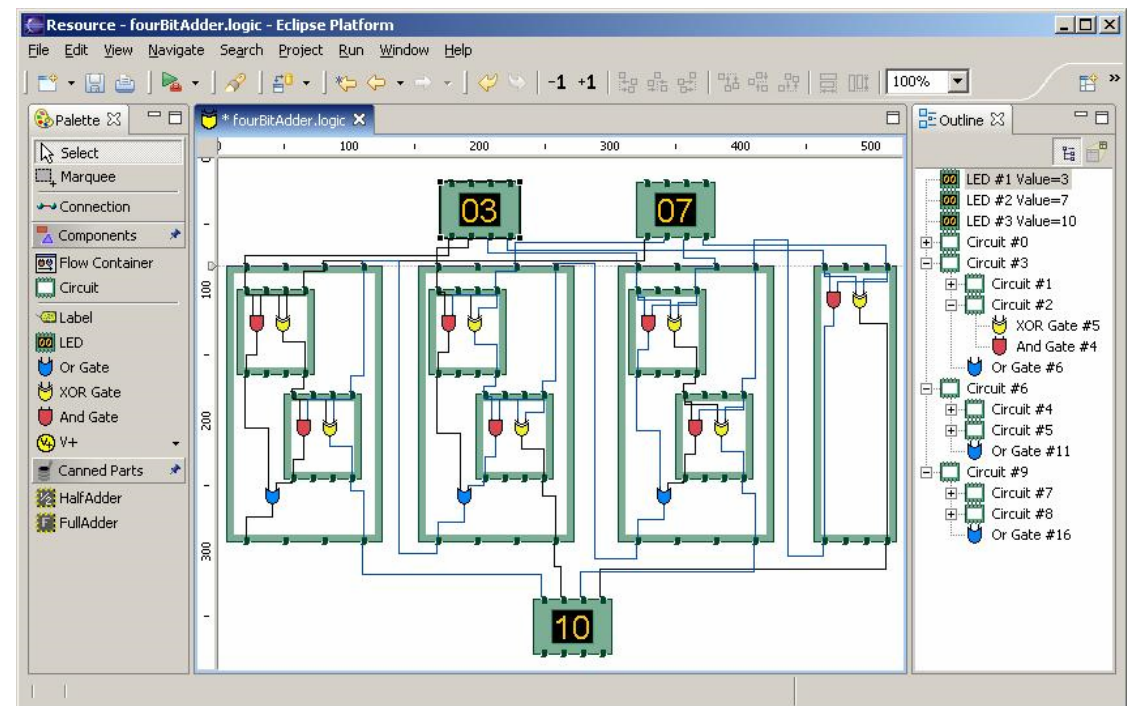
GEF (Graphical Editor Framework)

⇒ Framework for building rich graphical editors

- Draw2D - structured graphics drawing framework

- Graphical editor framework:

- MVC architecture
- Undo/Redo support
- Palette and common tools for manipulating objects
- Integration with Properties and Outline view



Eclipse Help

⇒ **Help UI on top of an extensible help content model**

- HTML and XML based system
- Context-sensitive (F1) help
- Dynamic content generation
- Search engine
- Dynamic-aware
- Highly scalable (used on ibm.com)

Help UI
User interface and dialogs

Help Core
API to access the documents

plugin.xml

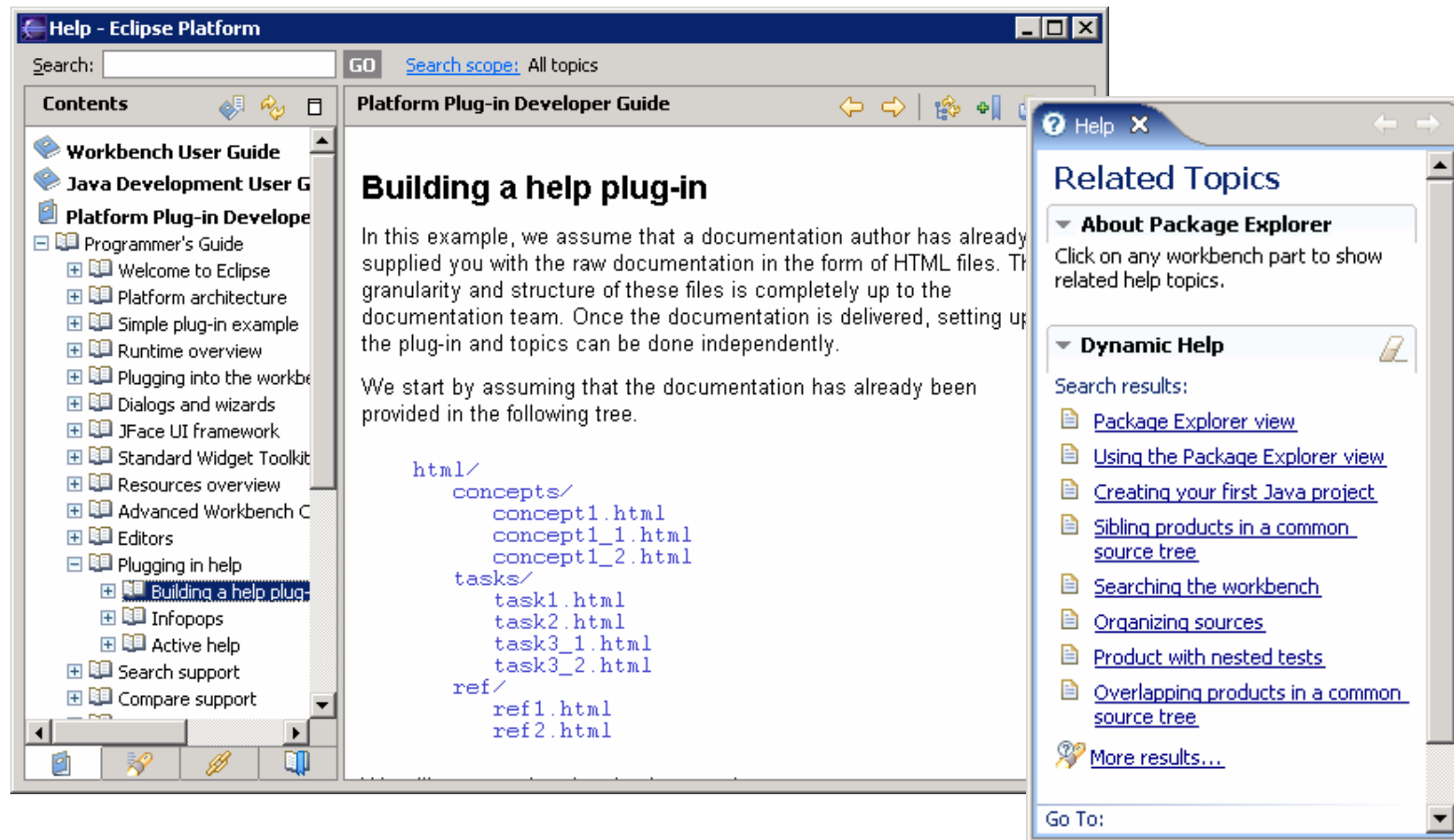
```
<extension point="org.eclipse.help.toc">  
  <toc primary="true"  
    file="doc/guide.xml"/>  
  <toc file="doc/tipsAndTricks.xml"/>  
</extension>
```

guide.xml

```
<toc label="Go wild user Guide">  
  <topic label="Getting Started">  
    <anchor id="gettingstarted"/>  
  </topic>  
  <topic label="Commands"  
    href="doc/cmds.html"/>  
</toc>
```

User Assistance

Eclipse Help – UI



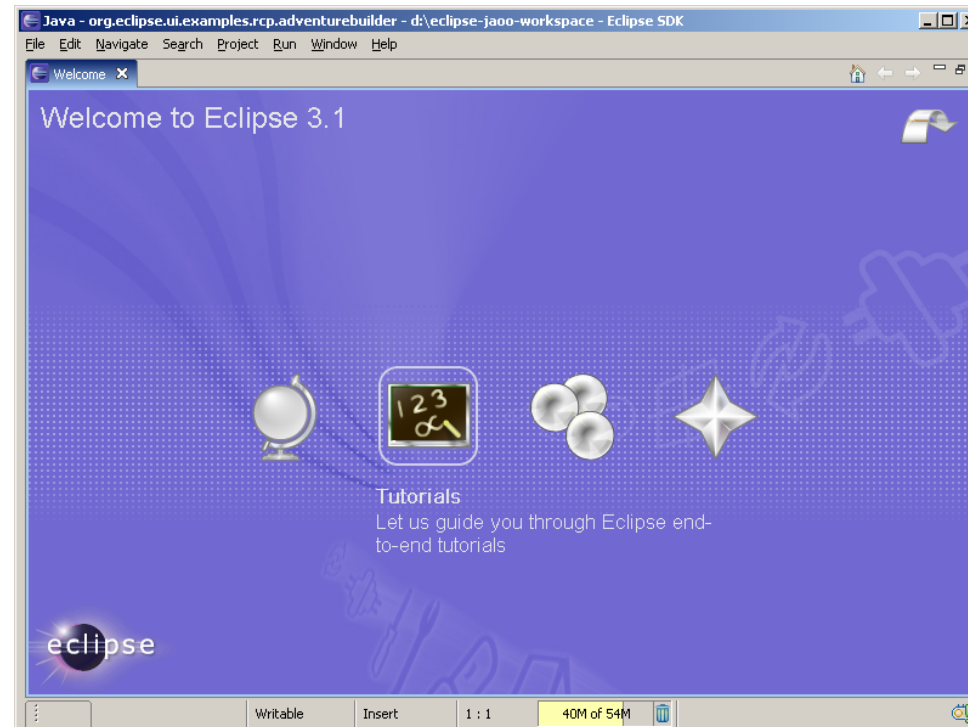
User Assistance

Intro support

⇒ Provides the “welcome experience” for your product

- HTML / CSS or SWT / Forms based
- Can run actions to drive the UI

Full mode:



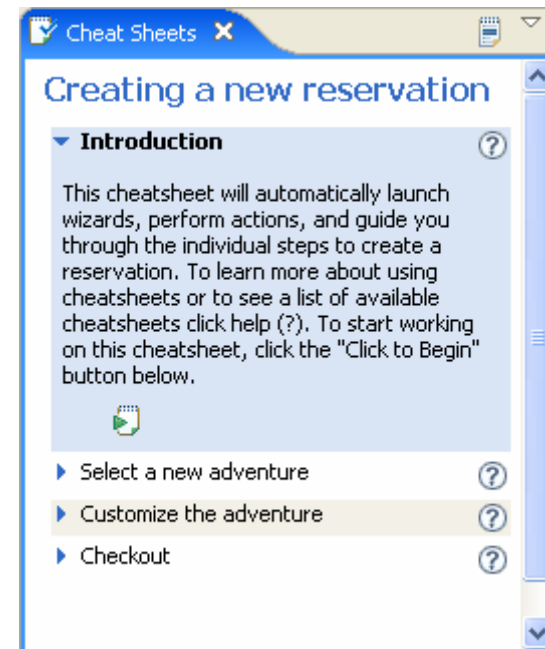
Standby mode:



Cheat sheets

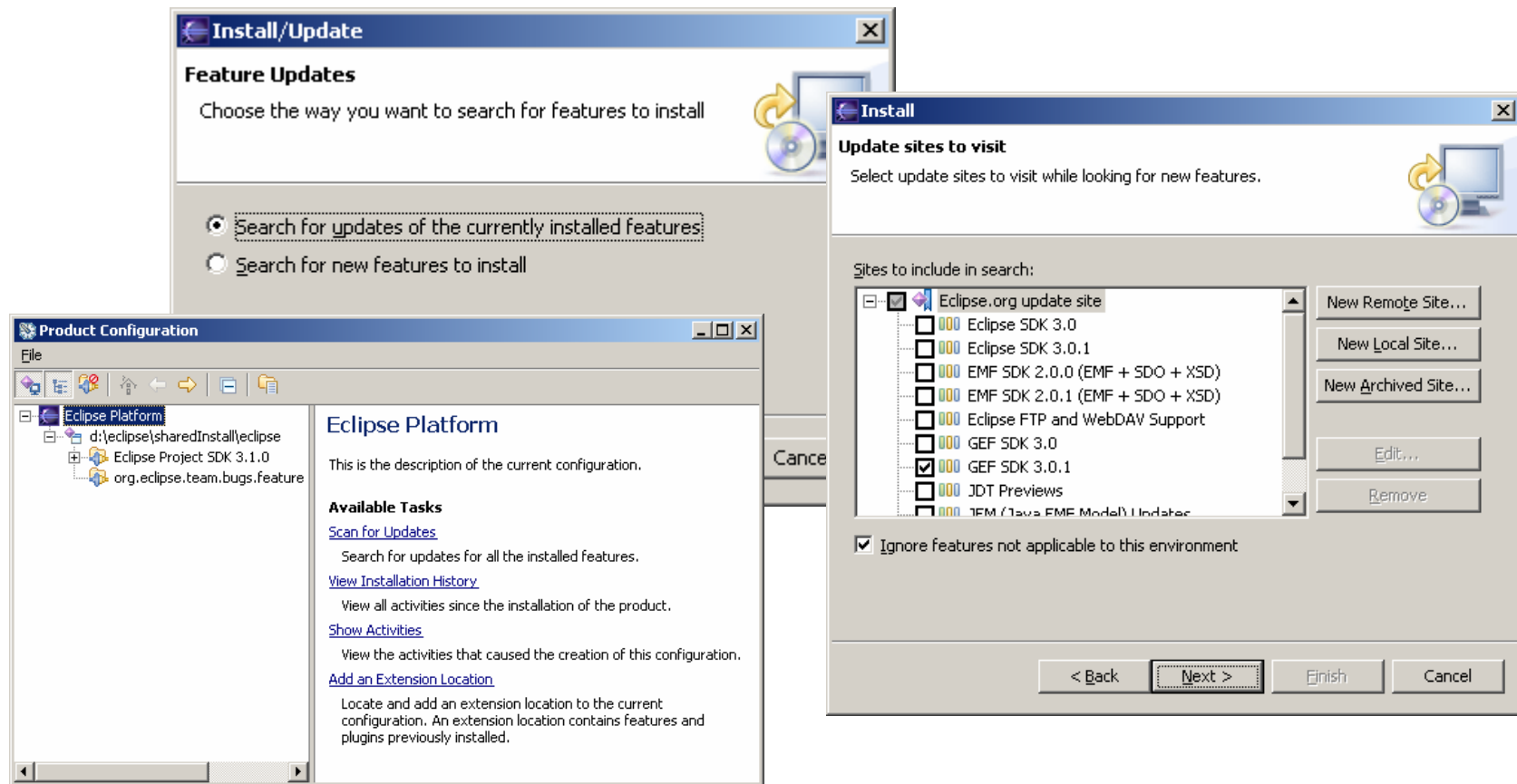
⇒ **Guides the user through a series of complex tasks to achieve a goal**

- Content written in XML
- Can run actions to drive the UI



Deployment and Update

Updating your application – Update UI

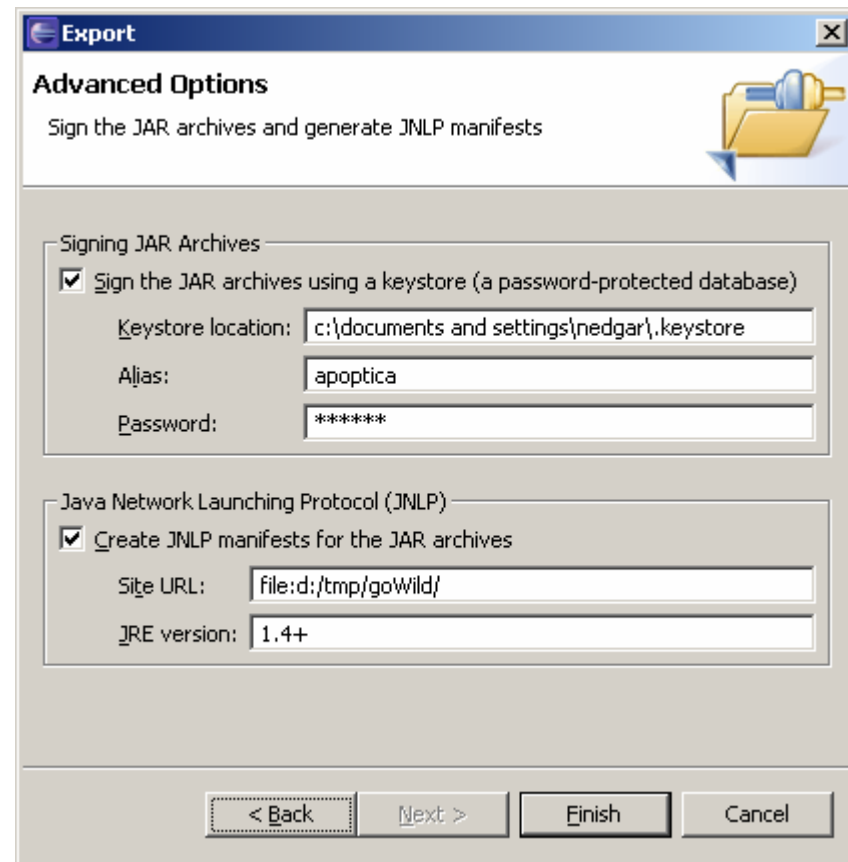


Deployment and Update

Eclipse and Java Web Start

⇒ RCP apps can be Java Web Start'ed

- PDE Feature Export helps with JAR signing and creating JNLP manifests



EMF (Eclipse Modeling Framework)

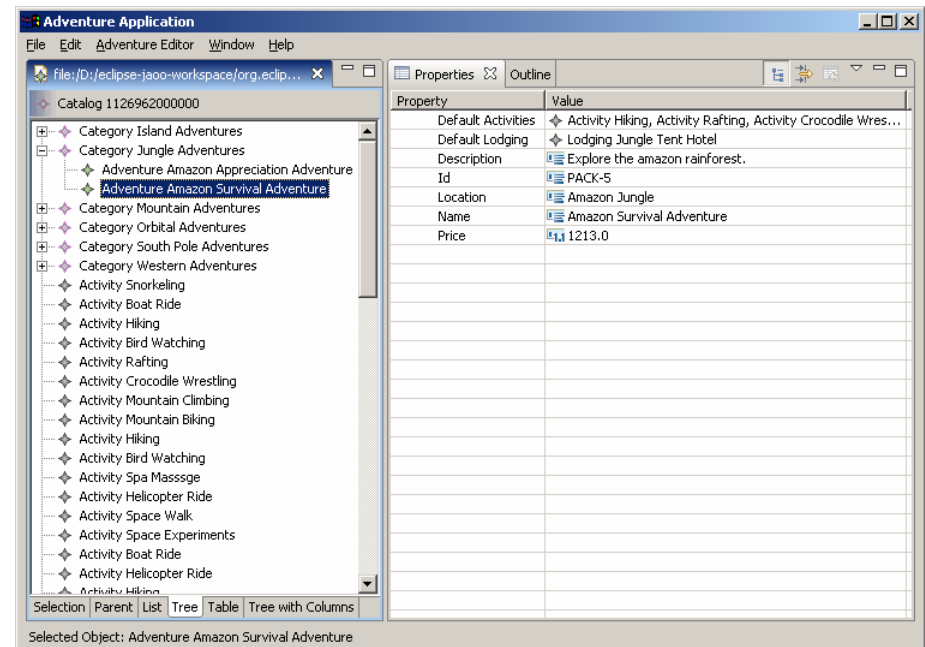
⇒ **Code generation from models**

⇒ **Supports model-driven development**

- Input models can come from:
 - by hand (using EMF model editor and Properties view)
 - from some existing design (e.g. UML from RAD or EclipseUML)
 - from annotated Java interfaces
 - some serialization schema (e.g. XSD, WSDL)
- generates efficient model code (incremental, non-destructive)
- efficient dynamic API for manipulating EMF objects and their metadata
- notification of changes to model objects (great for MVC)
- change recording and summarization
- serialization, e.g. to/from XML or XMI
 - lazy-loading of objects linked across different resources
- understands: relationship arity, inverse relationships, strong containment vs simple reference
- framework for model validation

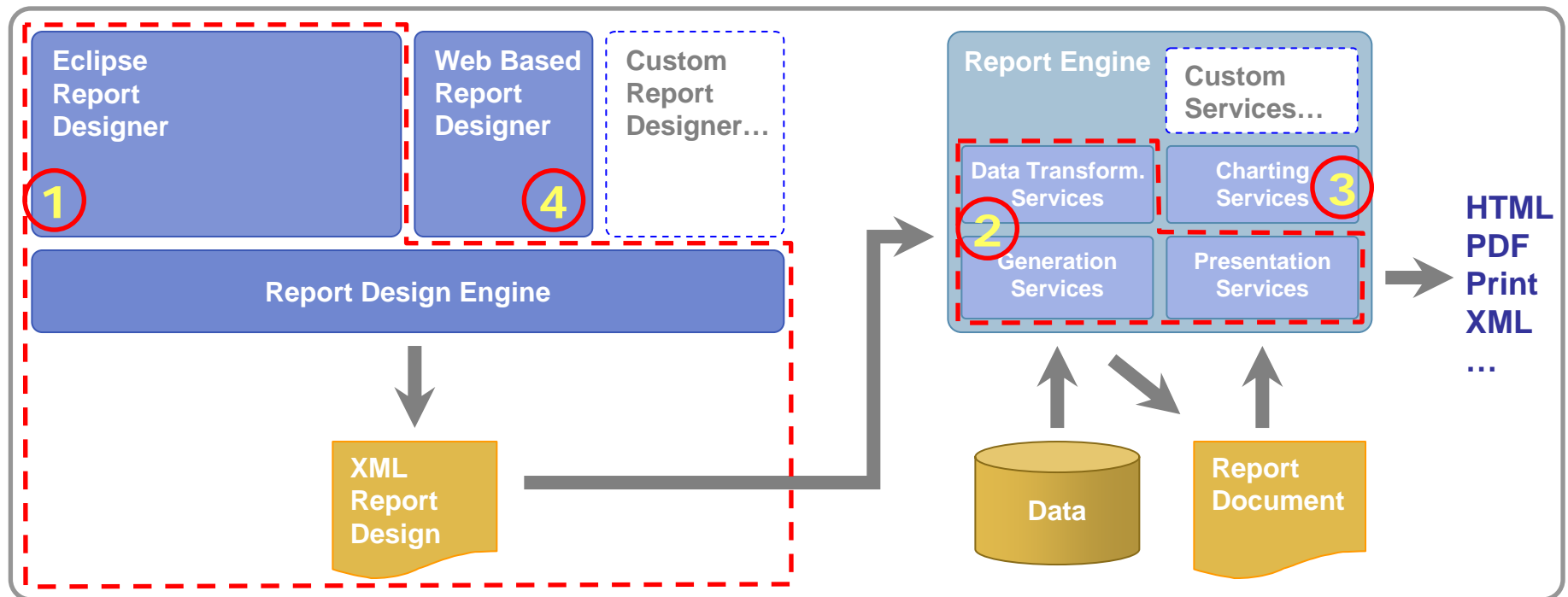
EMF (Eclipse Modeling Framework)

- can generate supporting adapters for:
 - showing EMF models in JFace viewers (with automatic update)
 - enabling command-based (undoable) editing
 - editing model objects in the Properties page
- can even generate a complete RCP app for editing instances of your model (good for rapid prototyping)



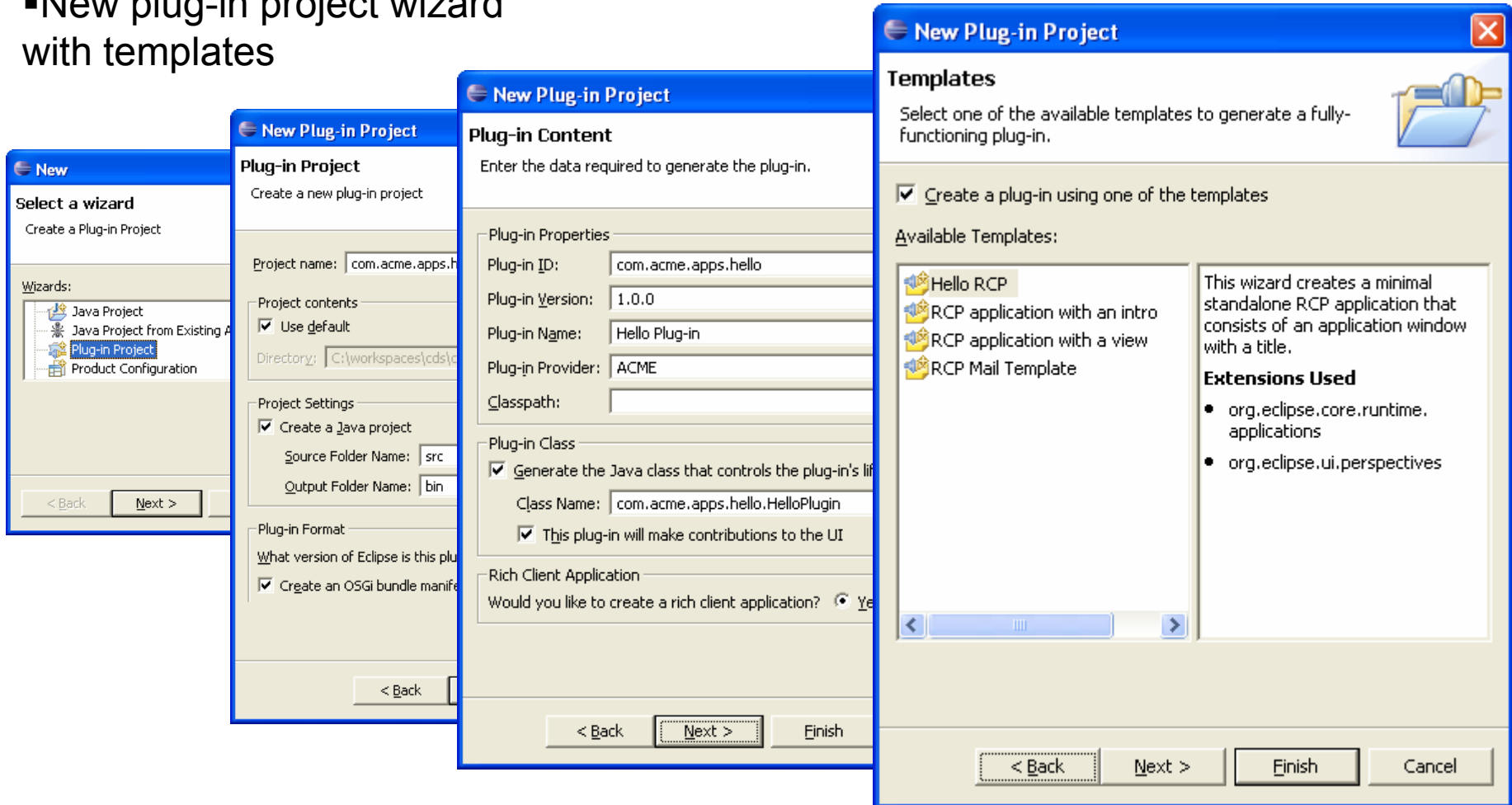
BIRT (Business Intelligence and Reporting Tools)

- Business Intelligence and Reporting Tools based on Eclipse
- Initially focused on embedded reporting for Java developers
- Proposal has 4 initial projects



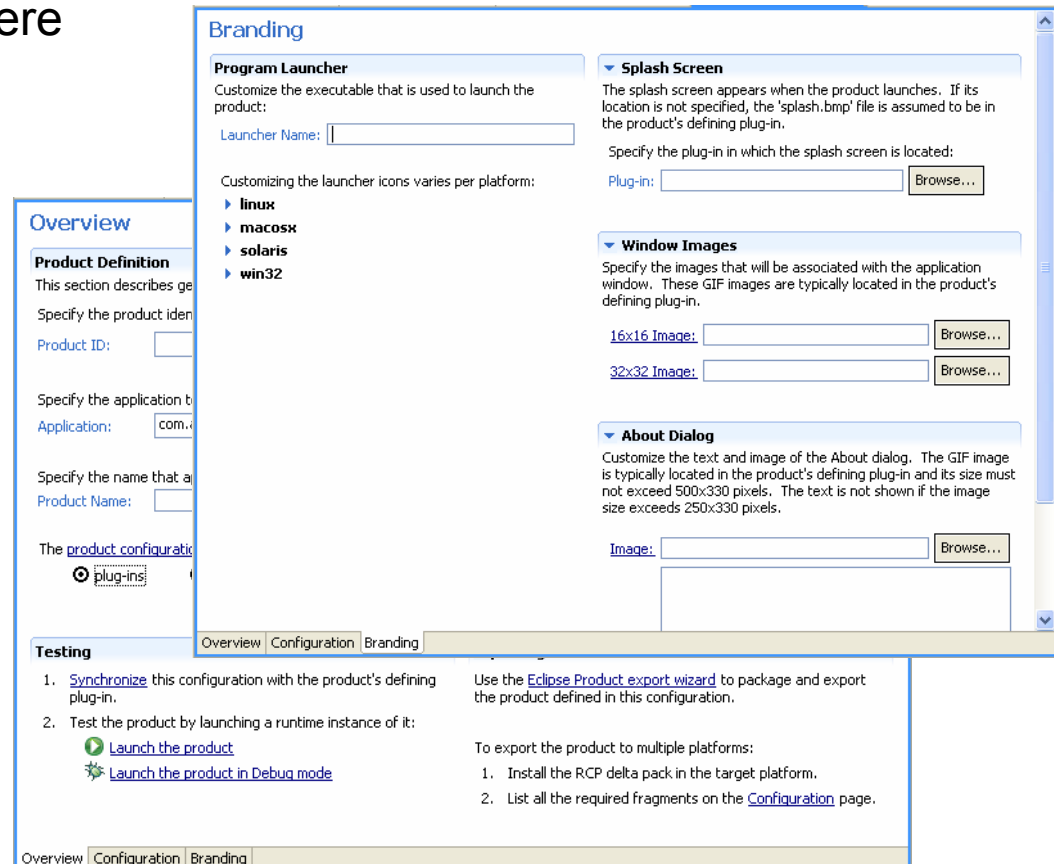
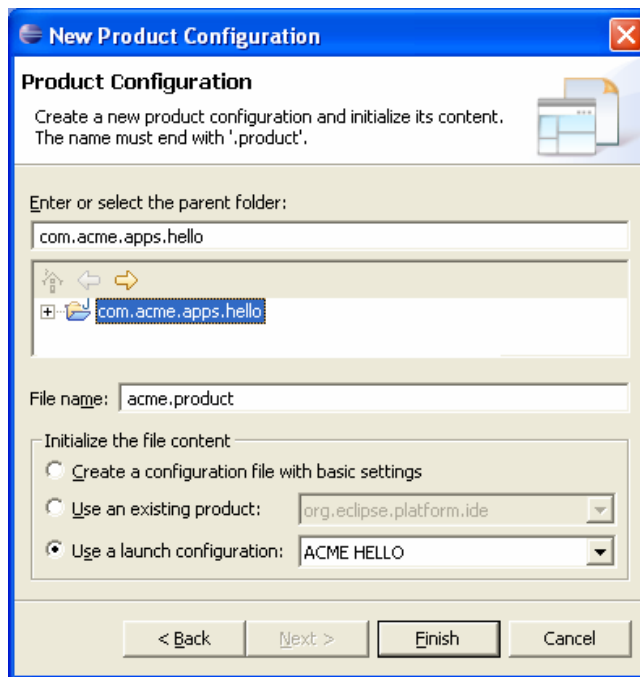
PDE (Plug-in Development Environment)

- New plug-in project wizard with templates



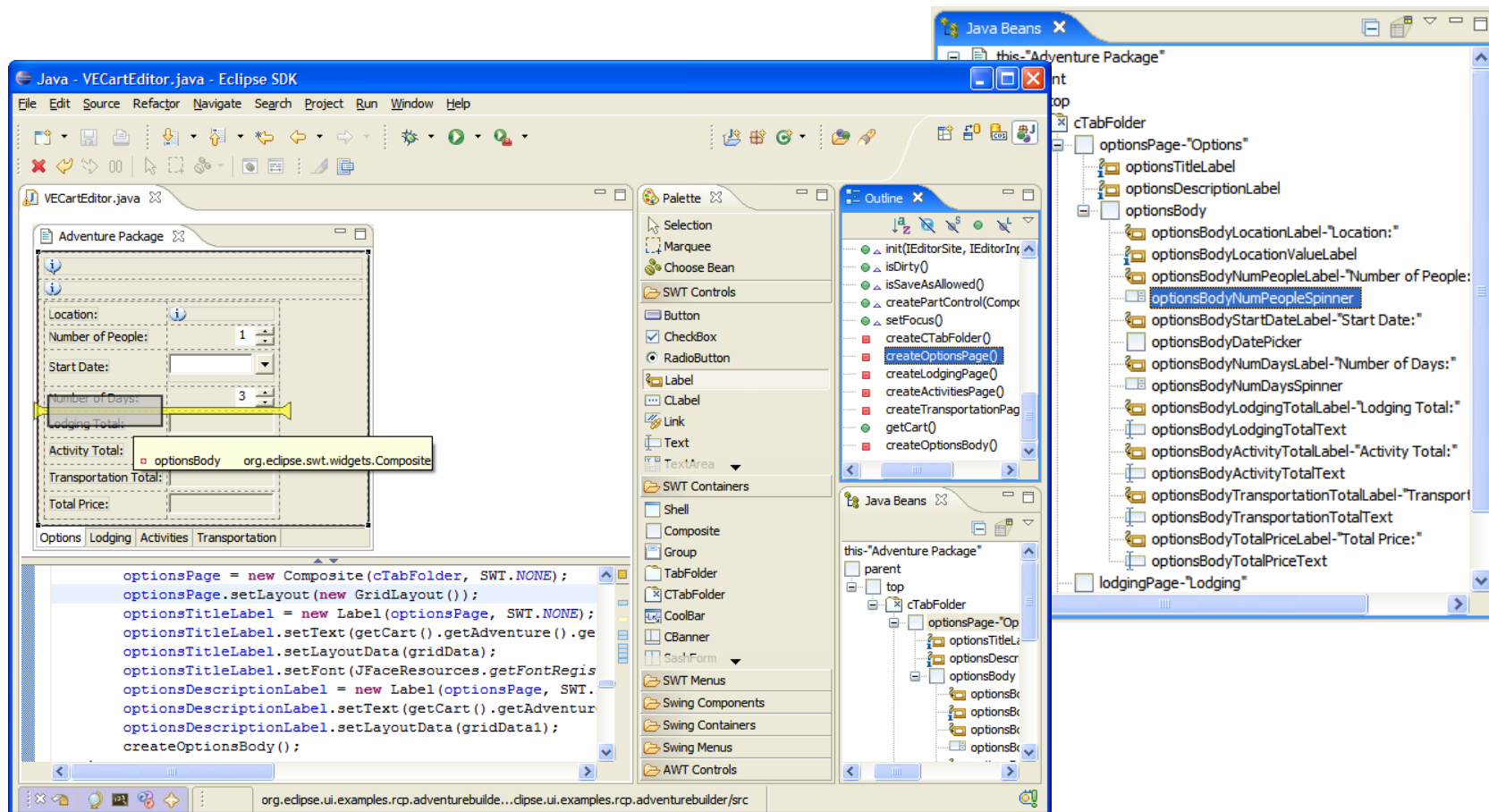
PDE cont'd

- Product configuration and branding editor
- Can easily test and export from here



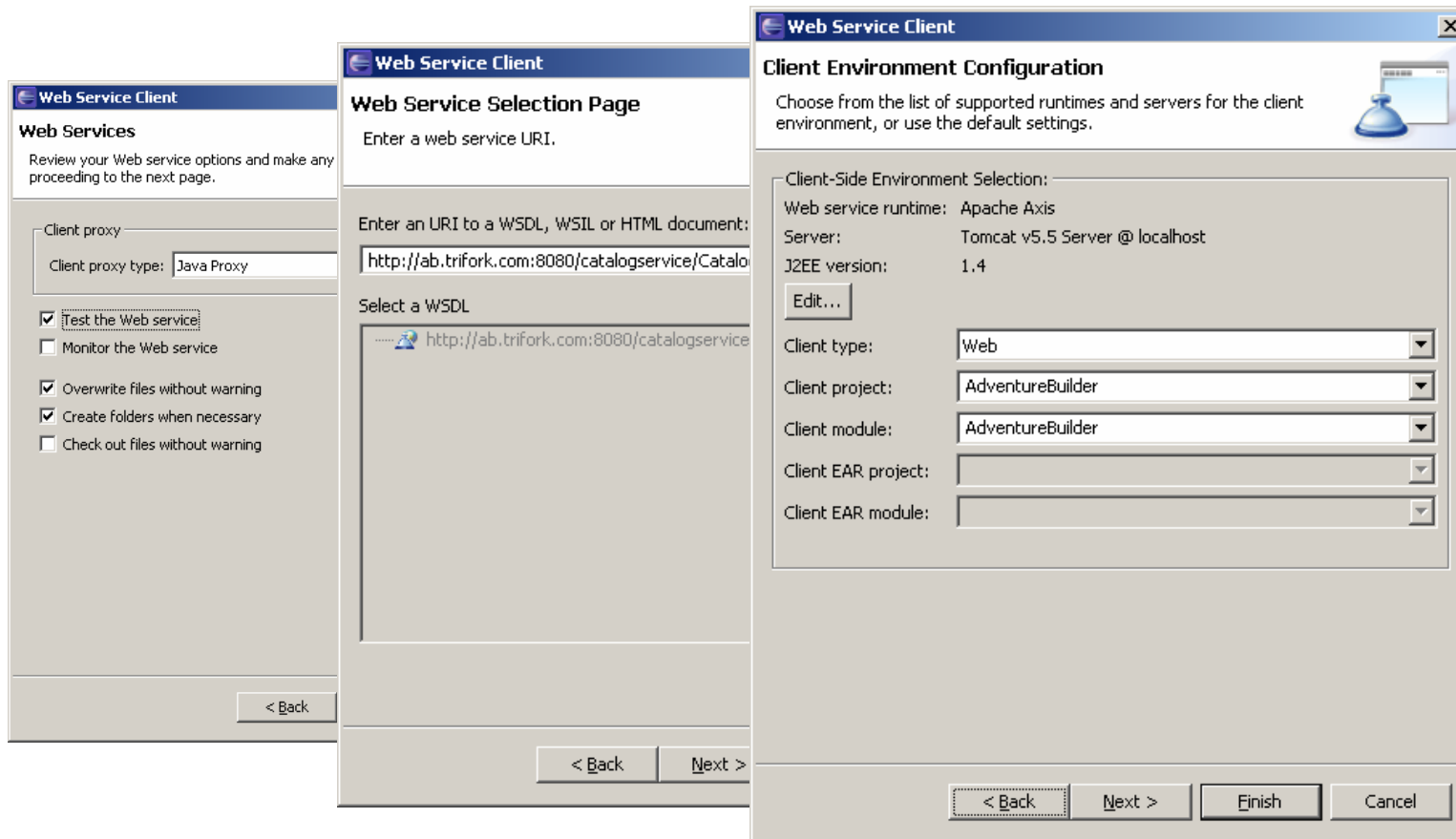
VE (Visual Editor)

⇒ GUI builder and framework for creating GUI builders



WTP (Web Tooling Project)

⇒ Tools for developing Web and J2EE applications

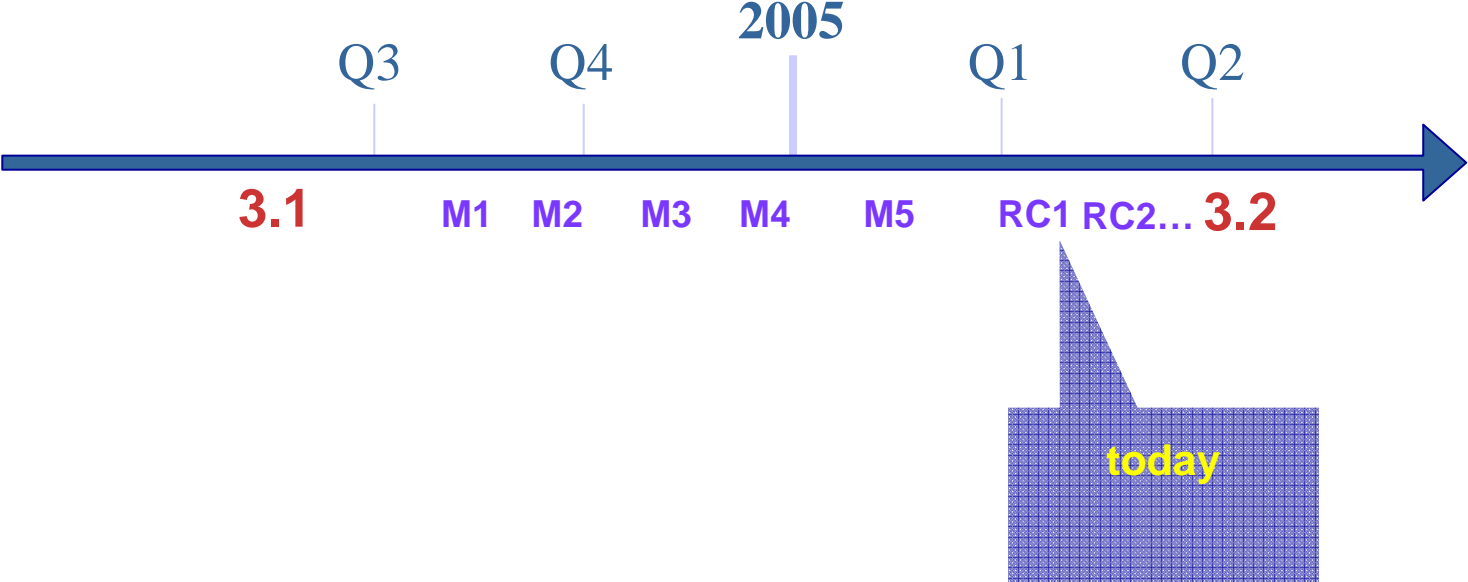


RCP summary

- rich set of functionality
- pervasive Plug-in architecture
- extensible
 - extensions and Extension Points
 - workbench provides many extension points
- scalable
 - supports large products (like RAD, Lotus Workplace)
 - scales down to embedded devices (eRCP, JCL/Foundation)
 - progressive exposure to functionality (perspectives, activities)
 - aggressive laziness
- customizable (see examples)
- dynamic
- interoperable: COM/OLE, AWT/Swing

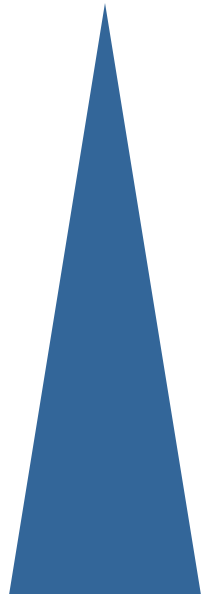
Middleware for Applications

on the way to 3.2...



conclusions

eclipse perception changes



1.0 “eclipse is a Java IDE”

2.0 “eclipse is a general tooling platform”

3.0 “eclipse is a general application platform”

where can I find out more ?

- www.eclipse.org
- RCP UI page:
<http://www.eclipse.org/platform> > UI > RCP Home Page
- Ed Burnette's RCP tutorials
- Platform and RCP newsgroups:
<news://news.eclipse.org/eclipse.platform.rcp>
<news://news.eclipse.org/eclipse.platform>
- Gamma, Beck: Contributing to Eclipse – Principles, Patterns, and Plug-ins, Addison-Wesley, 2004
www.awprofessional.com/series/eclipse