Building Applications for Multiple Platforms Using the Eclipse Rich Client Platform

Wayne Beaton, Eclipse Foundation, Evangelist

Jeff McAffer, IBM Rational, Eclipse RCP and Equinox Lead

(Martin Lippert, akquinet agile GmbH, Equinox Incubator Committer)

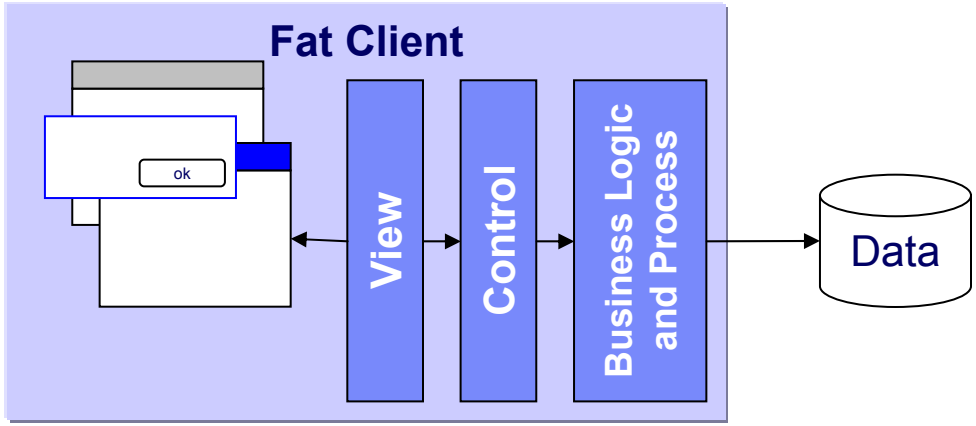
Agenda

- What is a rich client?
- What is Eclipse RCP?
- Equinox
- Building platforms
- Outlook

What is a Rich Client?

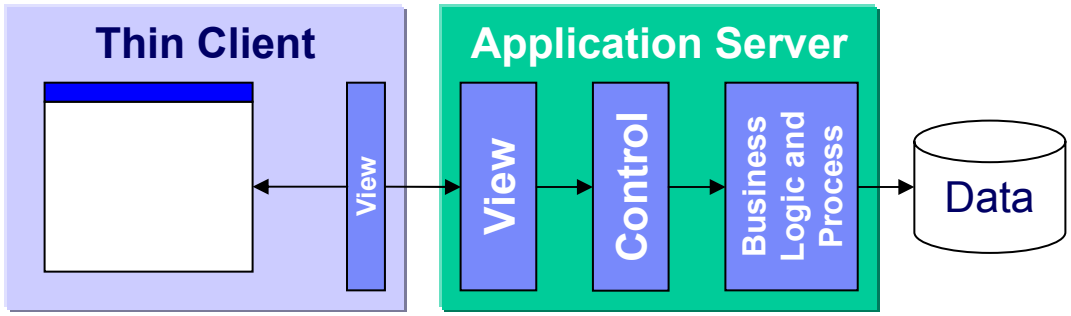
Evolution of the Client: Fat Client

- All application logic on the client
- Relatively difficult to update
- Rich user experience
- Monolithic application
- Platform dependent



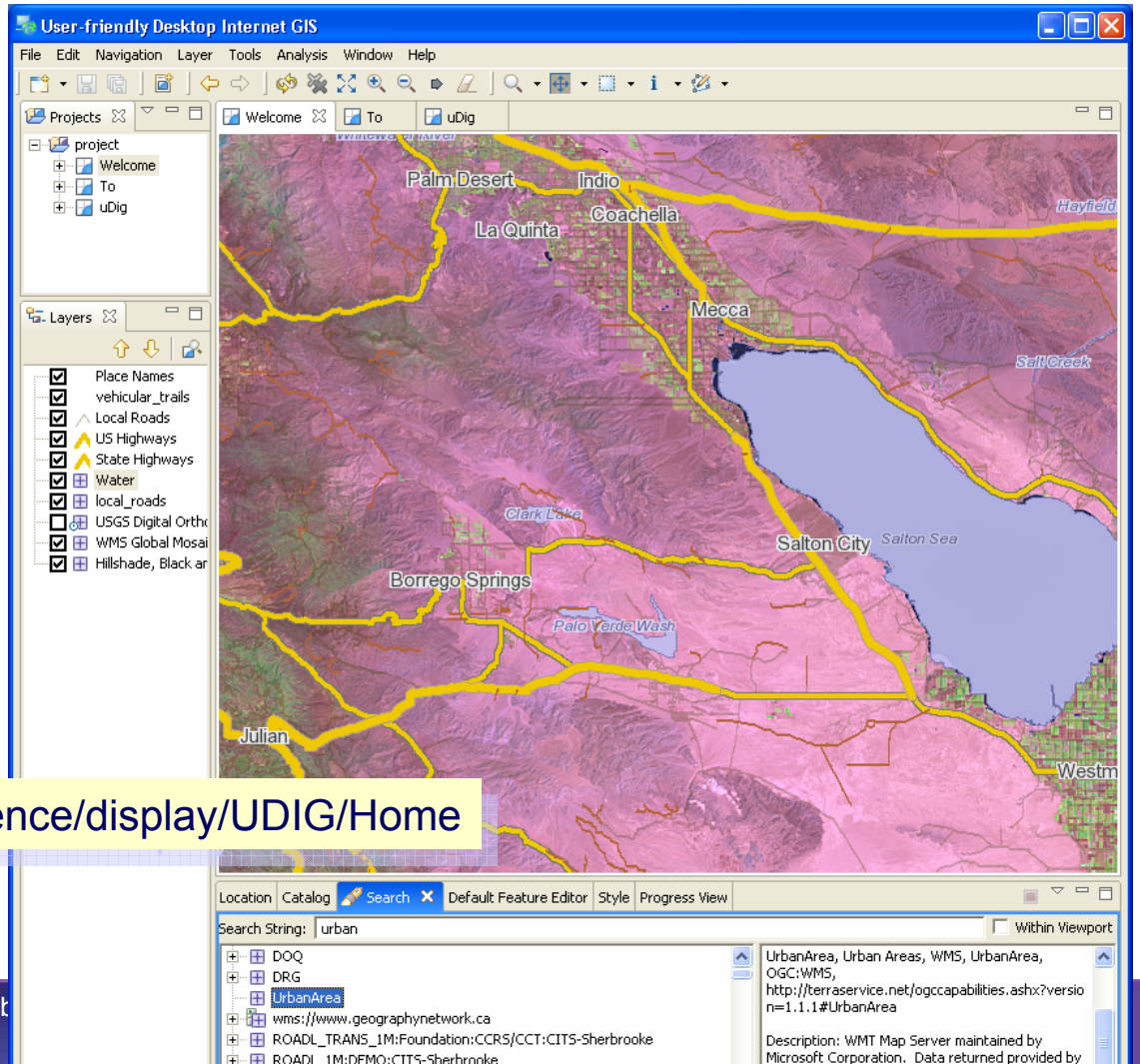
Evolution of the Client: Thin Client

- All application logic on the server
- Relatively easy to update
- Massively concurrent application
- Very simplistic user experience
- Client platform independent
  - Clients run in browser



What is a Rich Client?

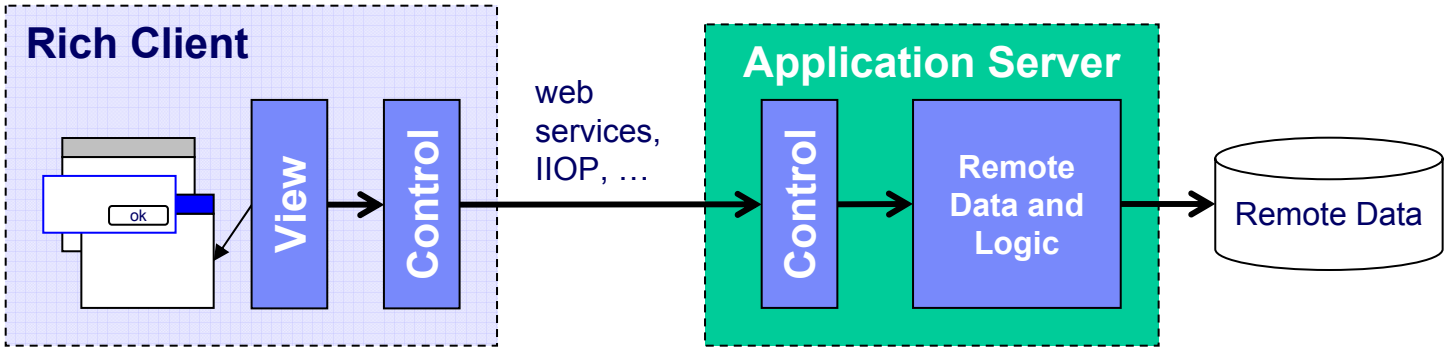
- An application that uses the windowing and GUI features of the operating system
  - Native widgets
  - Drag & drop
  - Integration with platform component model
  - Much, much, more…



http://udig.refractions.net/confluence/display/UDIG/Home

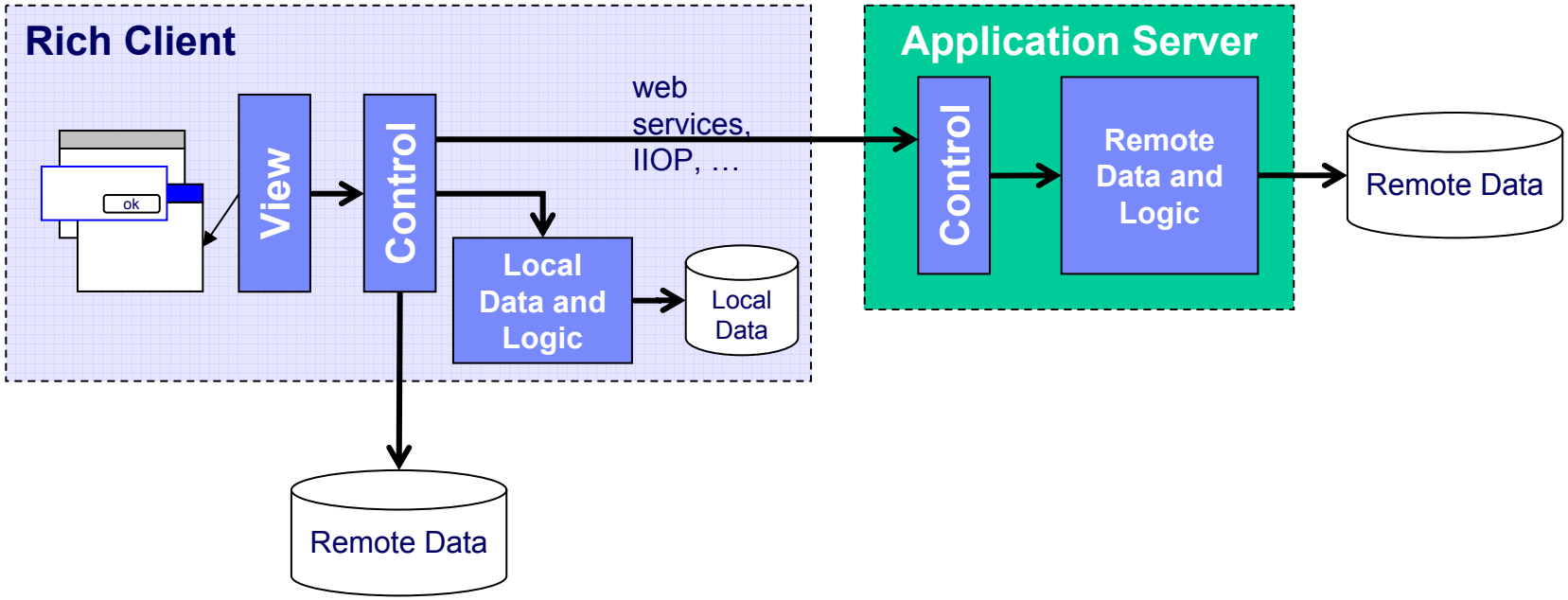
Three-tier Rich Client Application

- Some business logic stored remotely
  - Accessible through web service or other mechanism
  - WSDL-based web services, RESTful, IIOP, ECF, etc.



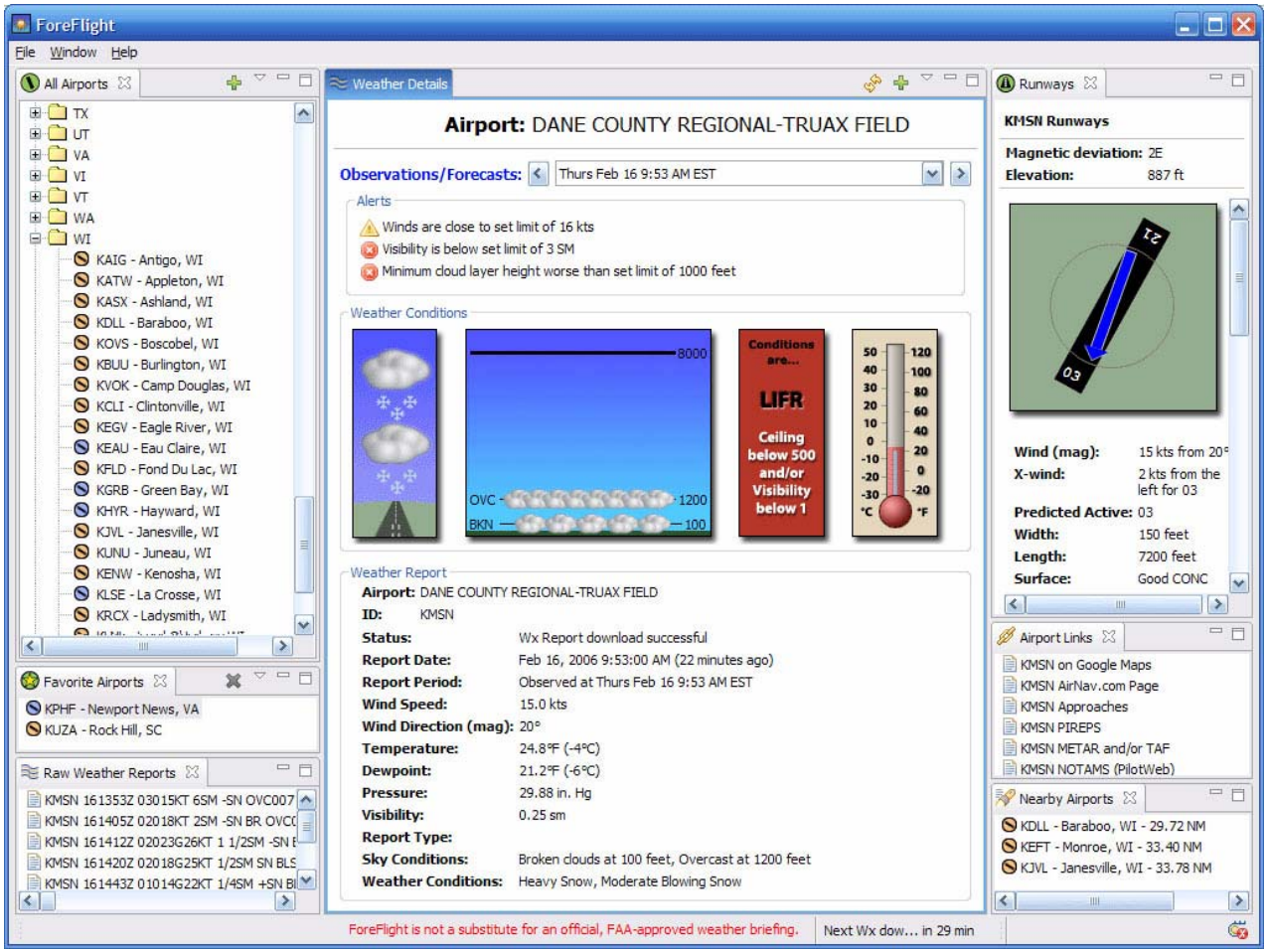
Combinations are Possible (and Likely)

- Access data from a variety of sources
- Cache remote data locally for offline work



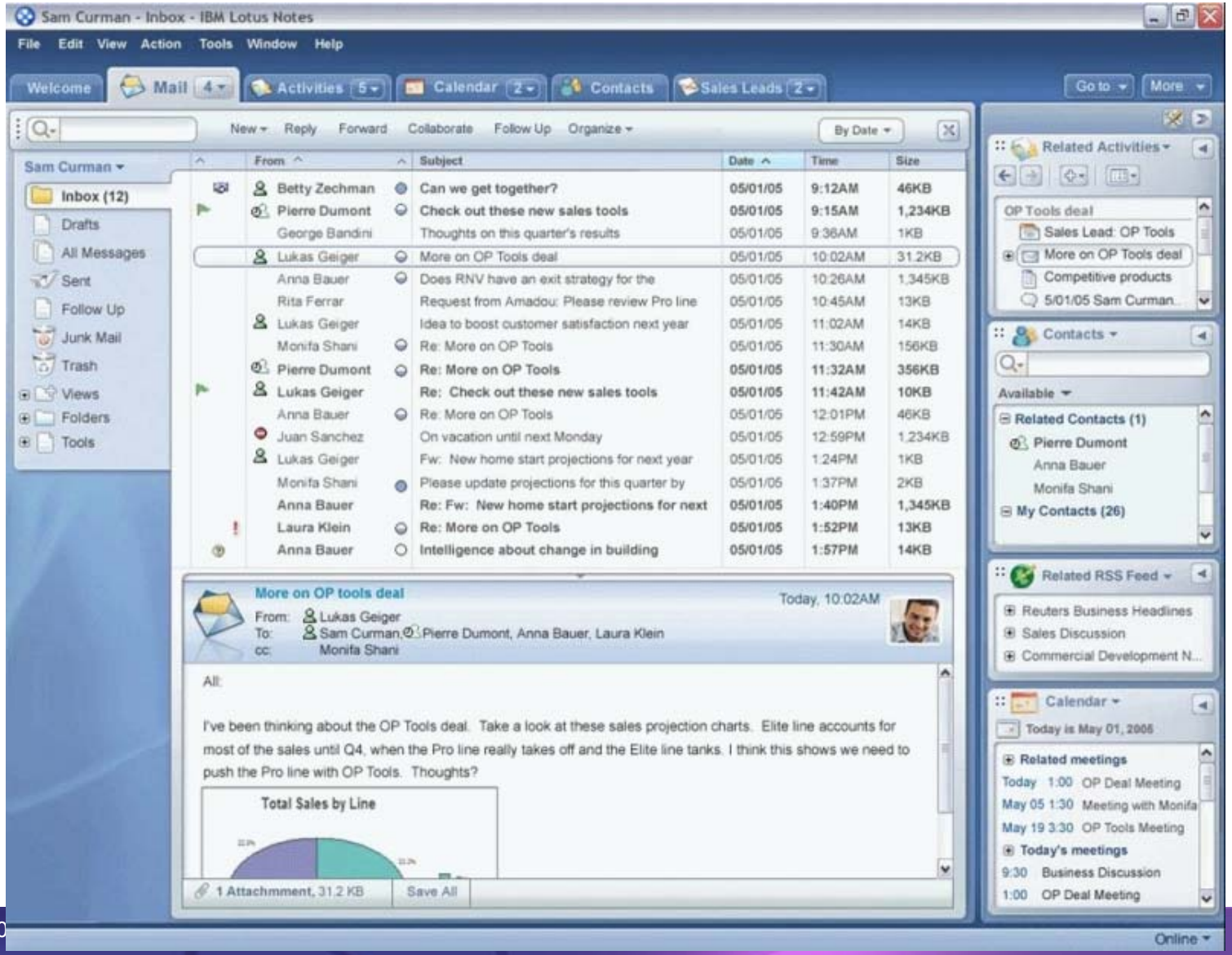
Example: ForeFlight

- Displays critical information graphically and prominently
- Displays alerts when conditions are near or exceeding the user's preferred limits
- Connects via the web to weather and information services
- Multiple ergonomic views of the weather that affects the go/no-go flight decision



http://www.foreflight.com/

Example: Lotus Notes “Hannover”



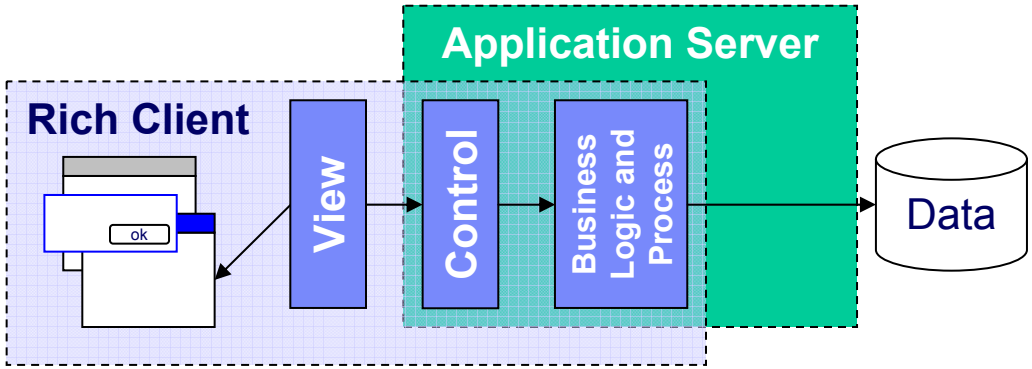
Example: RSS Solutions

- Advanced planning and scheduling (APS) solutions

http://www.eclipse.org/community/casestudies/RSSfinal.pdf

What is a Rich Client?

- Rich user experience
- Typically (though not necessarily) a client for some backend service
- Platform independent
  - Runs with little or no modification on multiple platforms and devices
- Component model
- Integrated update mechanism
- Extensible

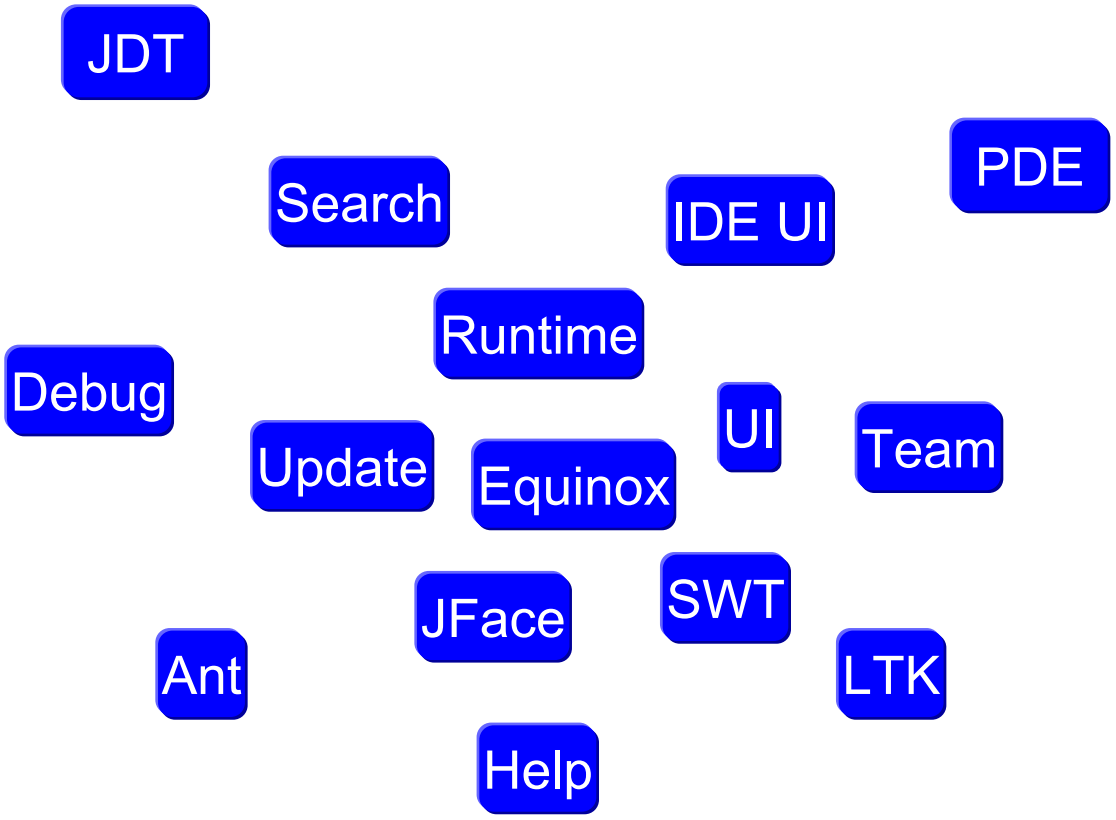


Rich or Thin Clients?

- Thin client
  - Casual users don't want to install software
  - Online banking, auction browsing
- Rich client
  - Power users demand responsive, rich, powerful user interfaces
  - Teller, bank manager, power auctioneer

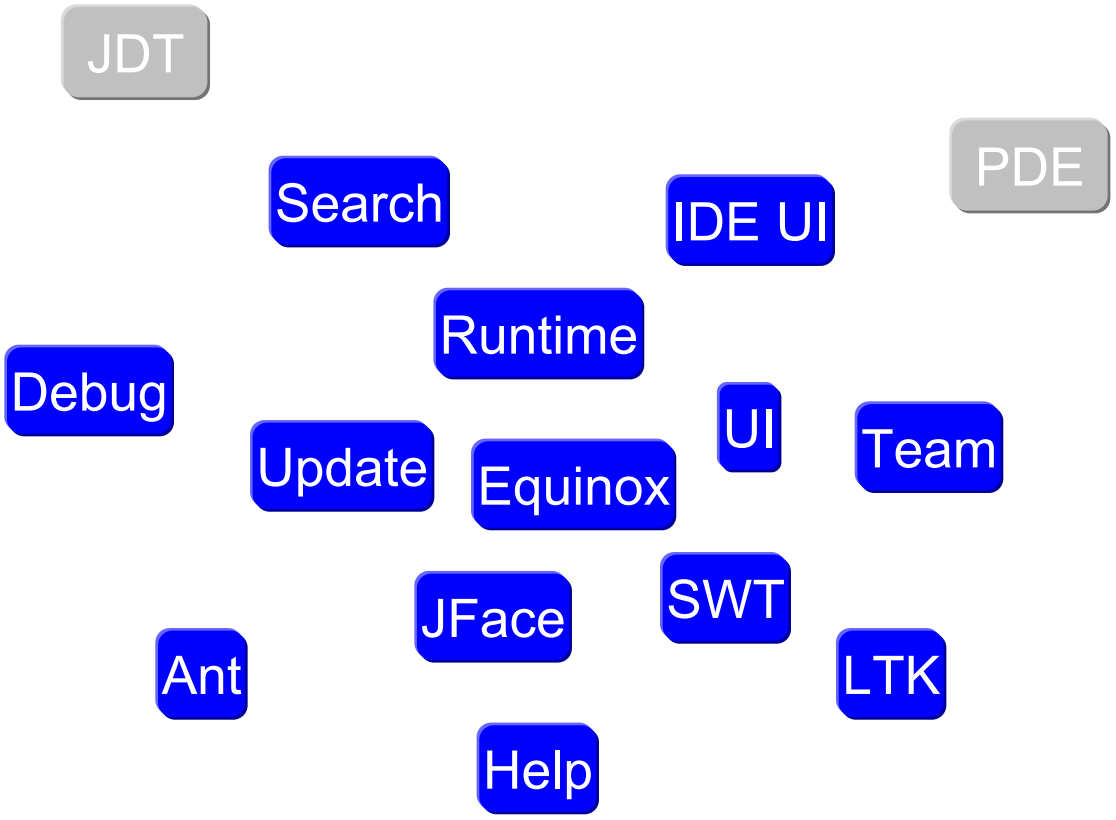
What is Eclipse Rich Client Platform?

Eclipse is a Composition of Components



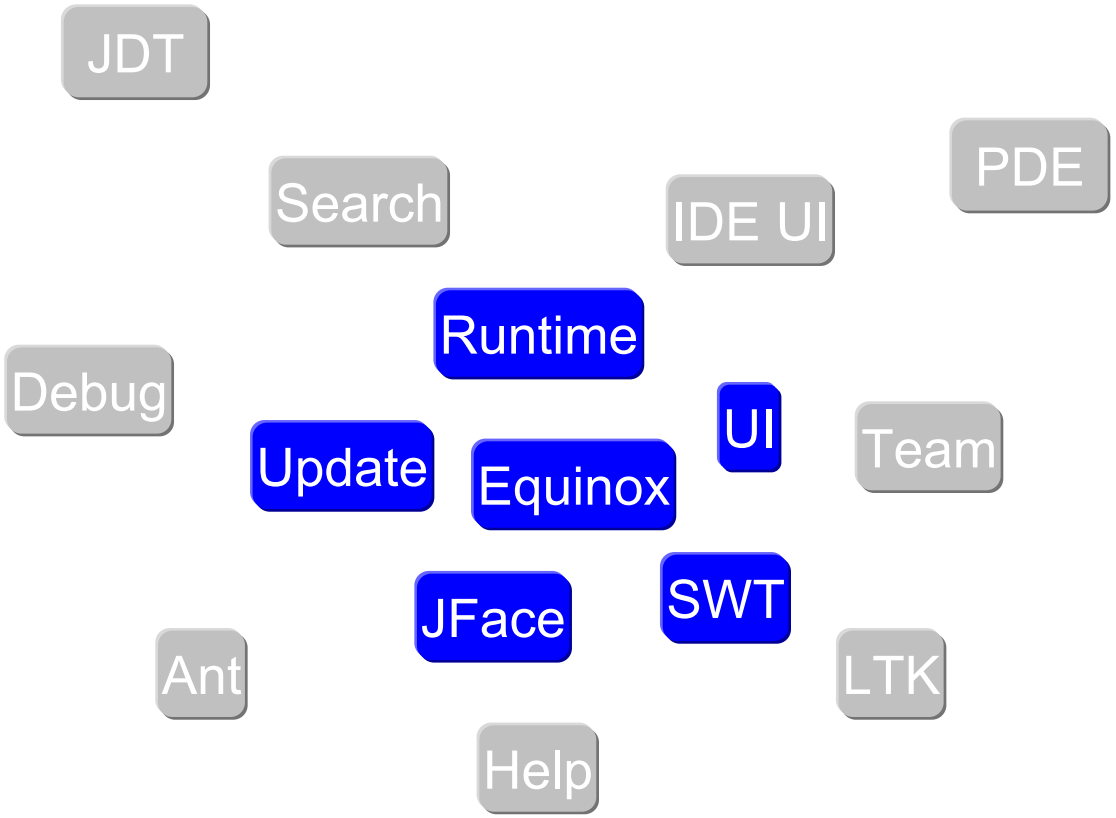
Eclipse SDK/Java IDE

Generic IDE Components



Eclipse IDE

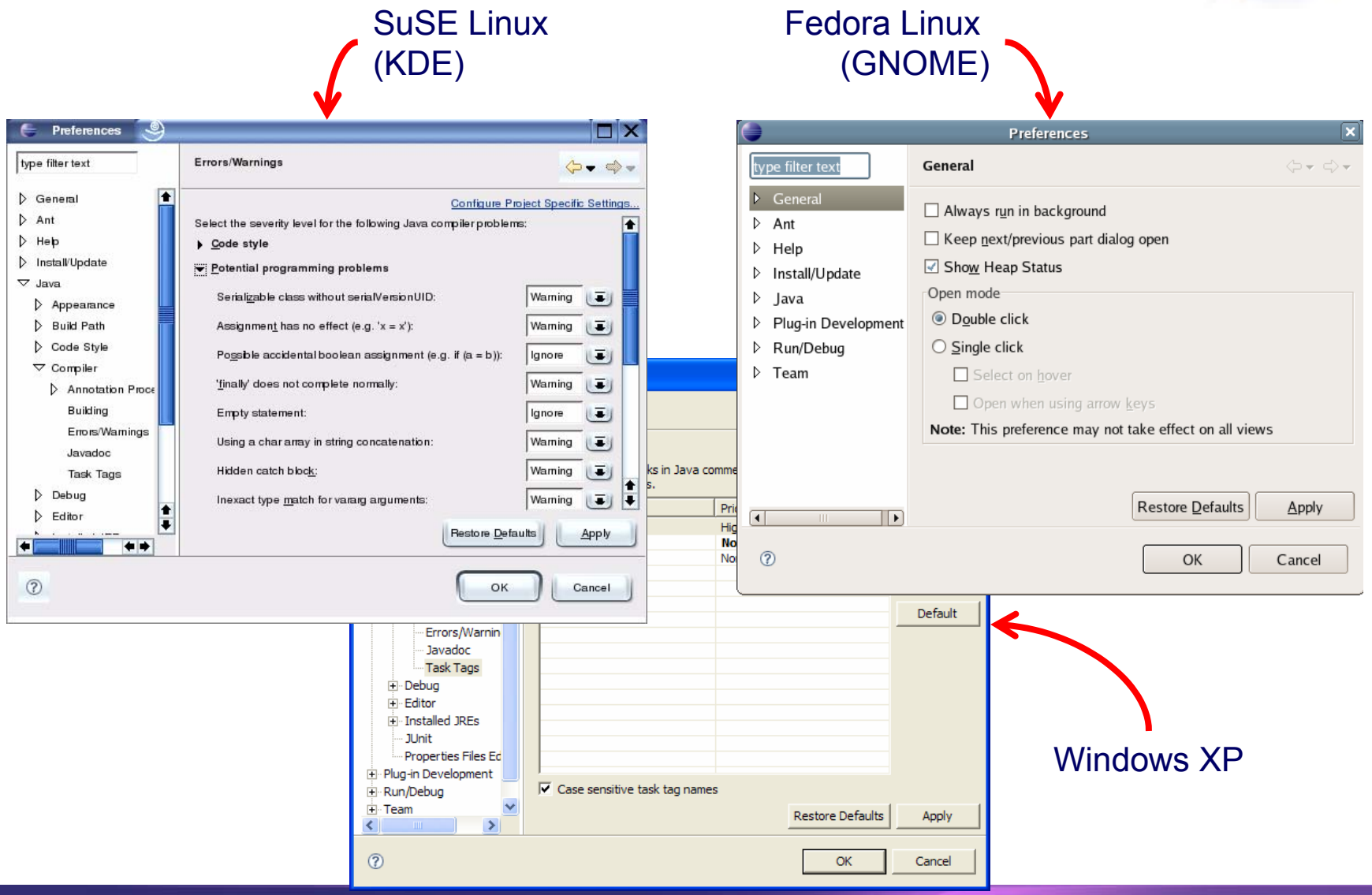
Eclipse Rich Client Platform



Why Use Eclipse Rich Client Platform?

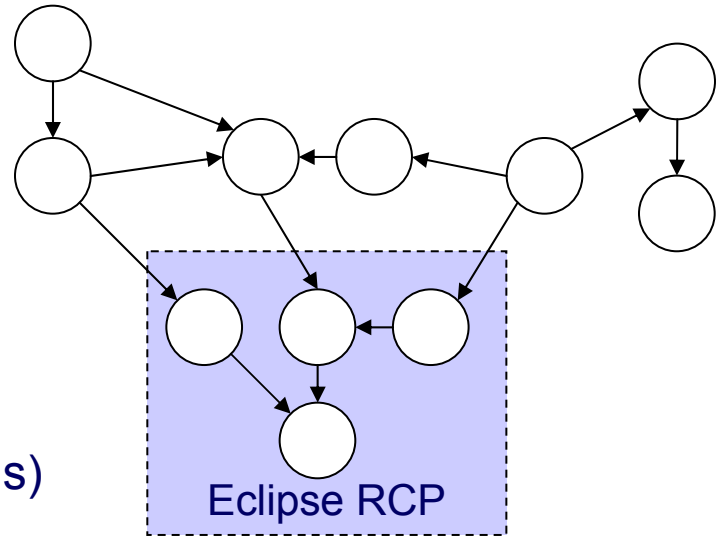
- A consistent and native look and feel across applications and features
- Provides common application services
  - Native look and feel
  - Window management
  - Standardized component model (Equinox)
    - Pervasive extensibility – Extension registry
    - Update Manager
  - Help system
- First-class development tools
- Middleware for building rich client applications!
  - Allows programmers to focus on core application not the plumbing
  - Don’t reinvent the wheel

Native Look and Feel



Equinox (1/2)

- Equinox is the Eclipse component model
  - Based on OSGi R4 specification
  - Standard Java lacks an explicit notion of components
- Components == Bundles == Plug-in
  - Versioned
  - Defined declaratively
  - Dynamically loadable/unloadable
  - Support dynamic update and install
- Explicitly define
  - Dependencies
  - Runtime visibility
  - Interactions (extension points/extensions)



Equinox (2/2)

- Components integrate without interfering
  - Required components explicitly set
  - Unrelated components do not have direct access to one-another
- Downstream components can access upstream components through the extension mechanism
  - Downstream component registers (declaratively) an extension point
  - Dependent components register (declaratively) extensions

Eclipse RCP as an Integration Point

- Integrating completely independent components is easy
  - Views from independent components can easily share the workspace
  - Menus populated by multiple components
- Integrating loosely coupled components requires planning
  - Sharing selection information
  - Drag and drop between views from different components
  - Extension point/extension mechanism for more intimate integration
    - Provide hooks to let downstream plug-ins participate
    - Not artificially restricted in any way

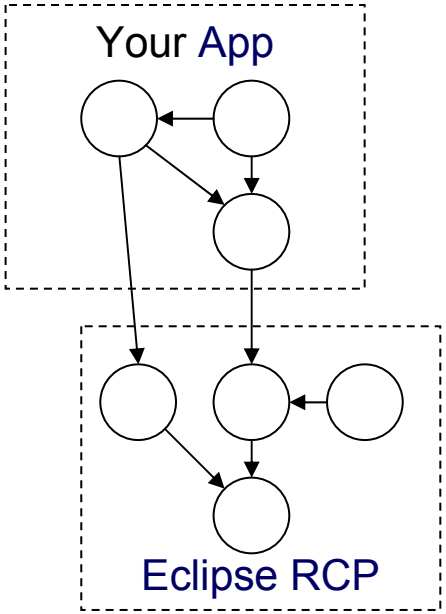
Trade offs

- Equinox provides the infrastructure to make components work
  - Architects and developers must factor the application into components
- Fine-grained components
  - Extreme example: one class per component
  - Reduced start up time
  - Easier updates and reuse
- Coarse-grained components
  - Extreme example: entire application in a single component
  - Generally easier to configure and maintain
- Balance
  - Best solution somewhere in between
  - Logical groupings of classes and resources

Building Platforms

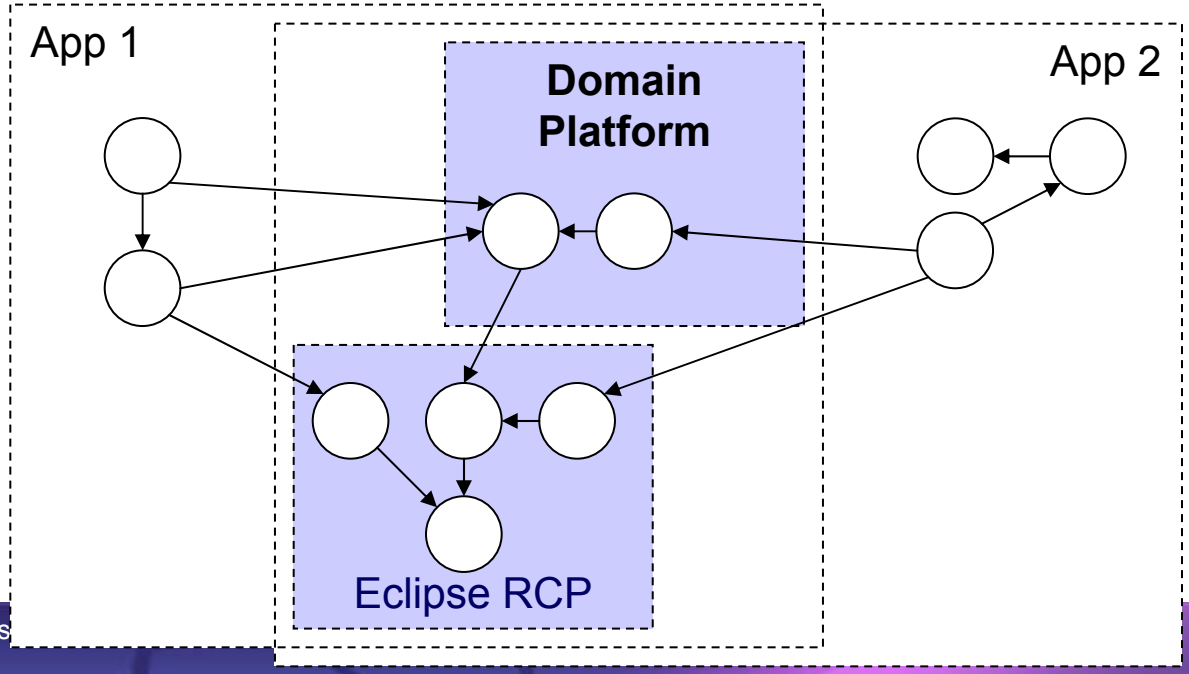
Building Platforms (1/2)

- It all starts with plug-ins
  - Eclipse RCP applications are composed of components that plug into the platform
- When starting development on RCP, it is common to provide a handful of domain-specific components that sit directly on top of RCP

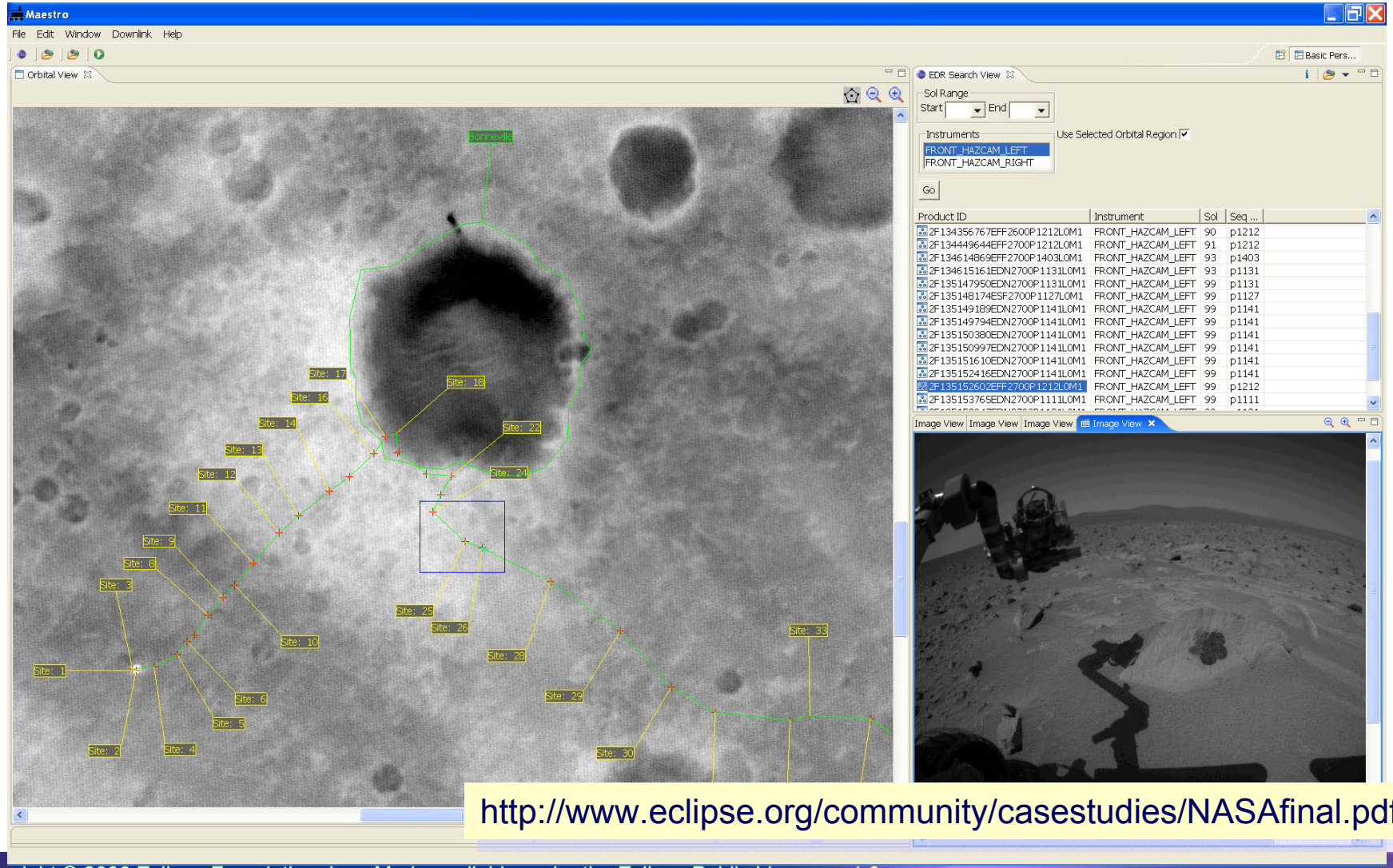


Building Platforms (2/2)

- It's natural for RCP development to spawn one or more "platforms"
  - A custom base for multiple development teams to build their applications upon
  - All of the examples in this presentation all have an underlying domain-specific platform



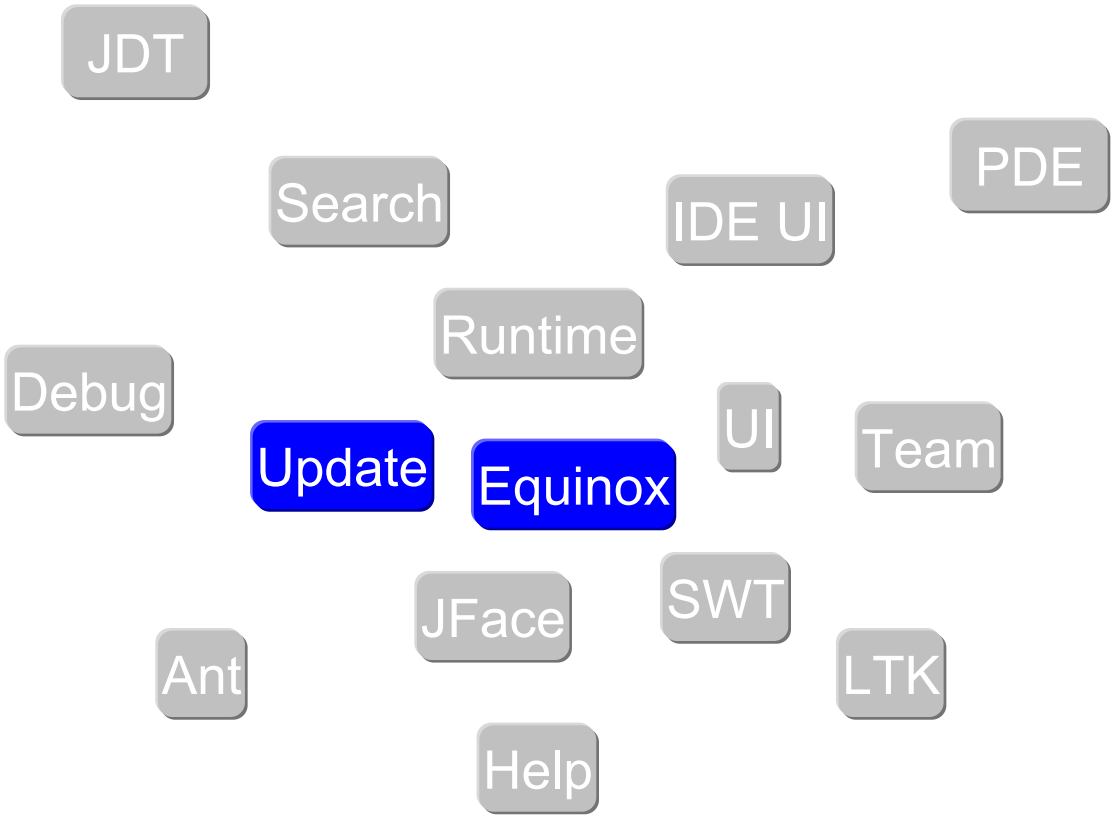
Example: Maestro – NASA Space Mission Management



http://www.eclipse.org/community/casestudies/NASAfinal.pdf

Outlook – Beyond client side applications

Eclipse Rich Server Platform (RSP)



Server-side Eclipse

- Why use the Equinox component technology only on the client side?

- Component model
- Update mechanism
- Extensibility

- All interesting for server-side applications as well
- Combination with Spring possible

Middle-tiers on Equinox

- Equinox can be used to implement middle-tiers
  - Same component model on both sides
  - Same extensibility for both sides
- Client and server could share the same components
- Integration with web-/app-servers possible

Equinox-based web apps

- Equinox can run inside a web app or the web-app can run on top of Equinox

- Web-app can be componentized
- Web-app can be designed and implemented for extensibility (Extension-Points)

Examples

- WAS 6.1
- Adobe Version Cue

- Apache Harmony
- Eclipse Rich AJAX Platform
- …

Recommended Reading

- Eclipse Rich Client Platform
  - By Jeff McAffer and Jean-Michel Lemieux
  - Addison-Wesley Professional
  - ISBN: 0321334612
- SWT : The Standard Widget Toolkit, Volume 1
  - By Steve Northover, Mike Wilson
  - Addison-Wesley Professional
  - ISBN: 0321256638
- Contributing to Eclipse: Principles, Patterns, and Plugins
  - By Erich Gamma, Kent Beck
  - Addison-Wesley Professional
  - ISBN: 0321205758