What's Next for Shared Editing?

- Shared Meta-Data
  - Markers (problems, tasks, bookmarks, etc)
  - Multi-user cursors, selection, awareness
  - Generated artifacts (parse tree, etc)
  - File Navigation
- Other requests
  - Auto-complete, quick fix, refactoring, etc
  - VCS integration
  - Remote searching (sharecode)
  - Specific Use Cases

Model Synchronization

- Graphical Operations
  - node position, connect, disconnect, etc
  - New synchronization/transforms for operations
- EMF Integration
  - EMF: Model creation and transformation
  - ECF: Distribution, model synchronization
- Model Synchronization
  - E4

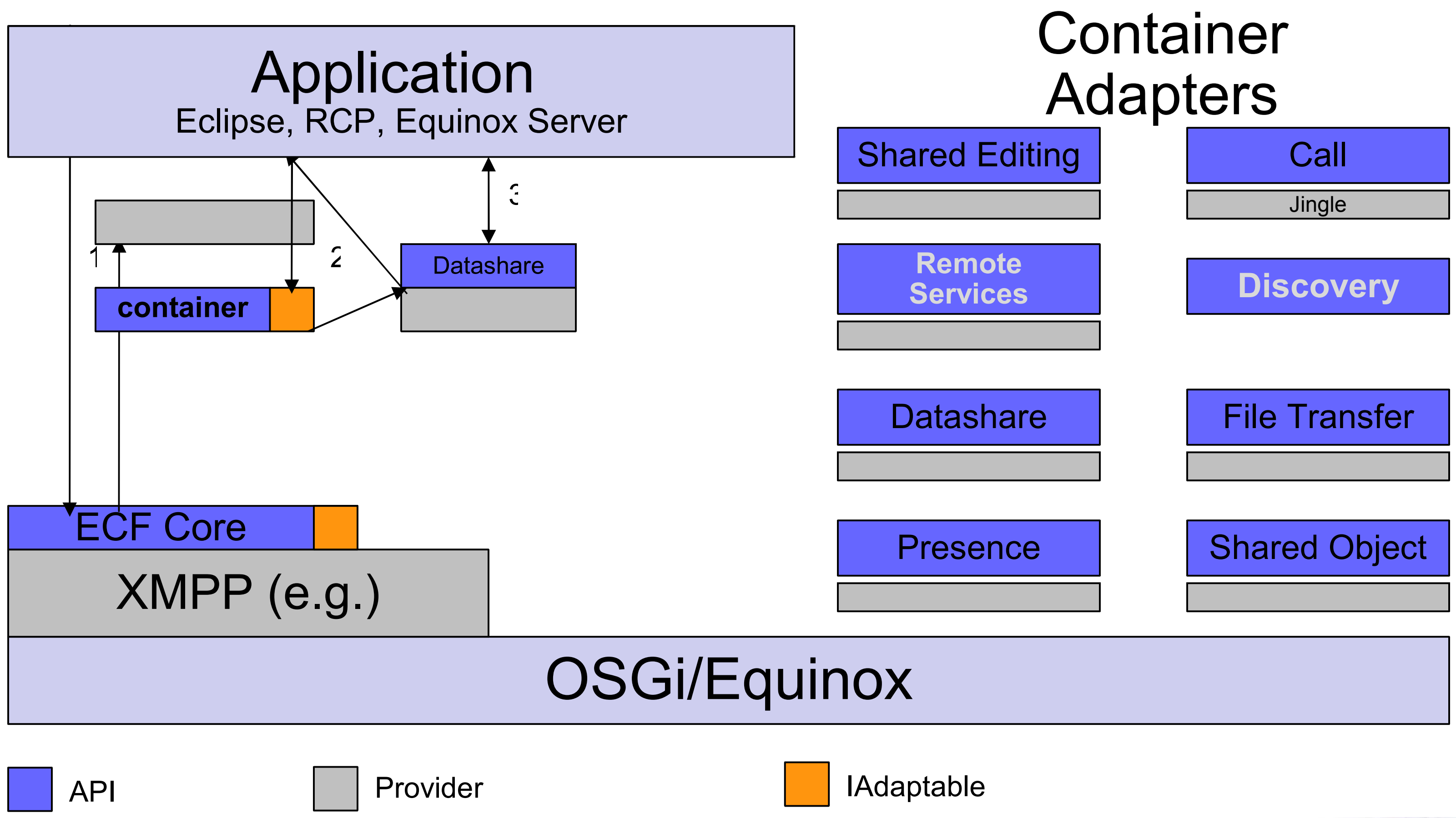
ECF$_1$: Integrated Team Collaboration

- Shared Editing
- IM/Presence/Chat
- Peer-to-Peer File Transfer
- Screen Capture, URL Sharing, View Sharing
- VOIP
- Mylyn Integration
- Workspace Sharing

ECF_2: Family of APIs

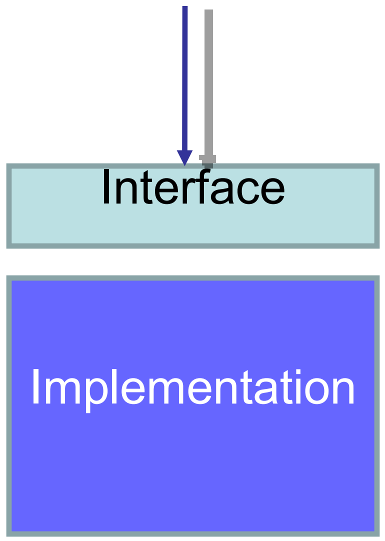
- Asynchronous messaging
- APIs as separate OSGi bundle
  - Modular API
  - Only use what's needed
- Provider architecture: Not protocol dependent
- API and Impl Extensibility
  - API: Adapters
  - Impl: Providers

ECF Architecture



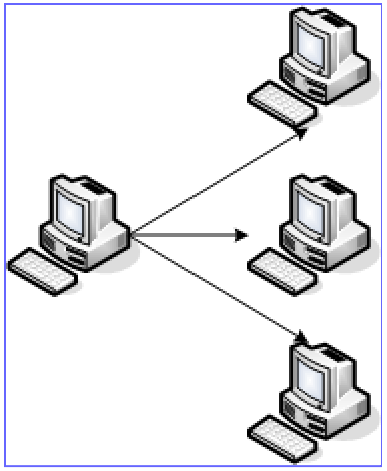
OSGi Services

- OSGi services provide
  - Encapsulation at a larger granularity
  - Loose coupling of functionality
  - Extensibility
  - Abstraction
- Remote services
  - Take this existing boundary to turn an application into a distributed application
  - Provide an abstraction to design distributed apps



OSGi services in the network

- Locate a service
  - Implementation for a given interface
  - Service discovery
  - Common knowledge
- Making use of a service
  - Providing service access via ECF API
  - “importing” the service into the local service registry
  - Providing a local service proxy

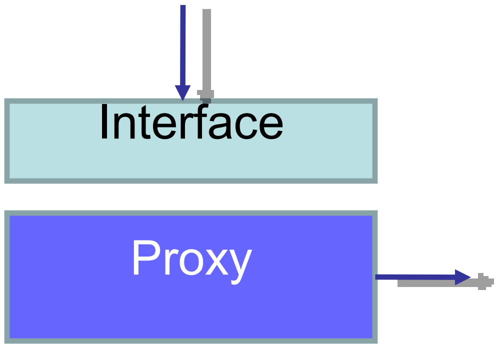


ECF service discovery – Overview

- Query for known/available services
  - Synchronous
  - Asynchronous: add/remove a service listener and get notified about service discovery/”undiscovery”
  - Query by filter/example (TODO)
- Manual and automatic service announcement

Remote Services

- OSGi services which cross address spaces
- Same ideas:
    - Ask for a service (-reference)
        - Can trigger service discovery
    - Get the service
        - Get a proxy for the service
        - Proxy generation can be proactive or reactive
    - Use the service
        - Method invocations become remote invocations



Transparent API

- Service and client remain untouched
- Some entity (not the client) states the demand
- Proxy is already present when the client asks for the service
- The service remains agnostic against distribution, as far as possible
- Seamless and flexible transition from local to remote services

Non-Transparent API

- Client is aware of distribution
  - Retrieve an IRemoteService object
  - Explicit app-level failure handling
- Explicitly call remote invocations
- Call semantics can differ from local service calls
  - One-shot invocation (non-blocking)
  - Asynchronous invocation
    - E.g., with listener callback
    - Futures

Contribute

- Use
  - Try It/Report Bugs/Request Enhancements
  - Fix it/extend it to your liking
- Join Project with Committers
  - Jump In and Work With Us/Community
  - Integrate with Google APIs and Services: GoogleTalk, Jingle, Calendaring, Google Groups, Social Networking APIs, Others
  - ECF Joining Runtime Project (server-side Equinox)

ECF Project Info

- Website: http://www.eclipse.org/ecf
- Wiki: http://wiki.eclipse.org/ECF
- Blog: http://eclipseecf.blogspot.com
- ECF 2.0 ~~will ship~~ is shipping with Eclipse Ganymede
- The work on ECF 2.1 has just started ☺