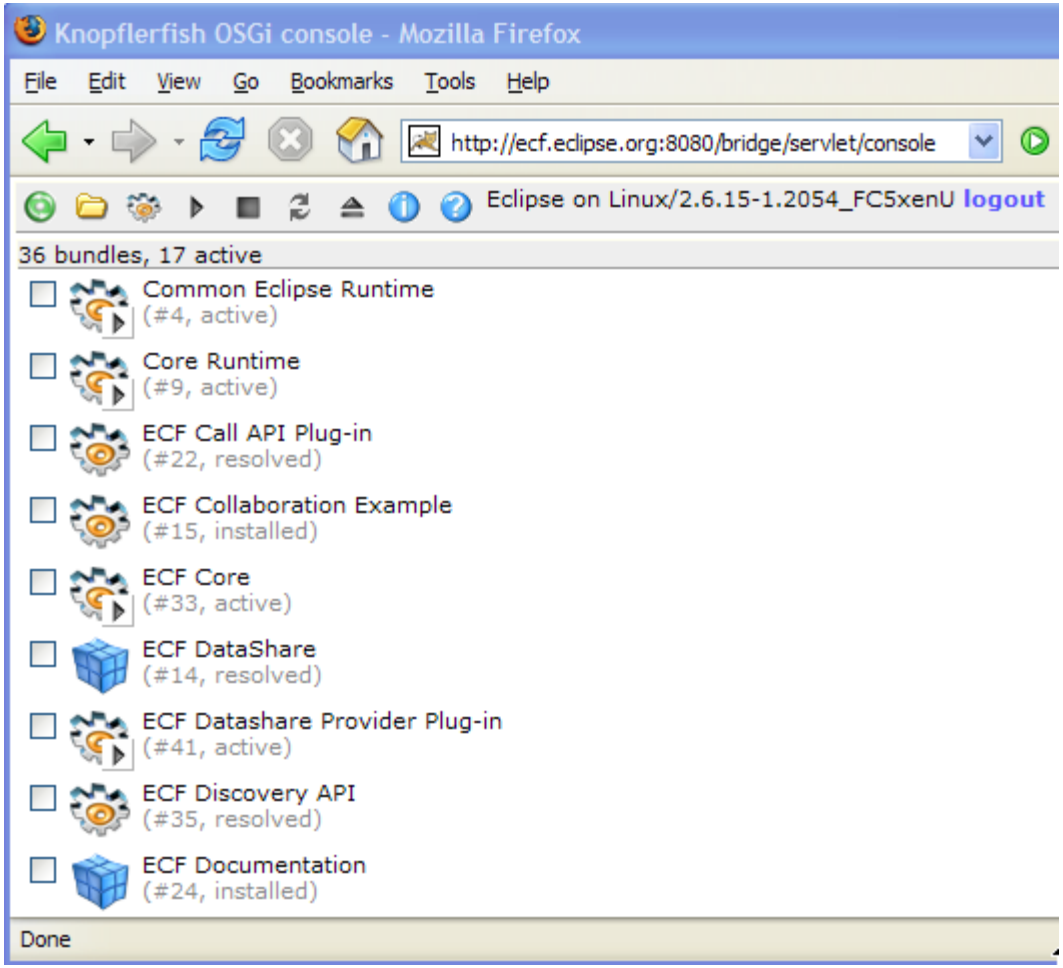
ECF on the Server

Equinox/OSGi + ECF = 'Equinox Service Bus'

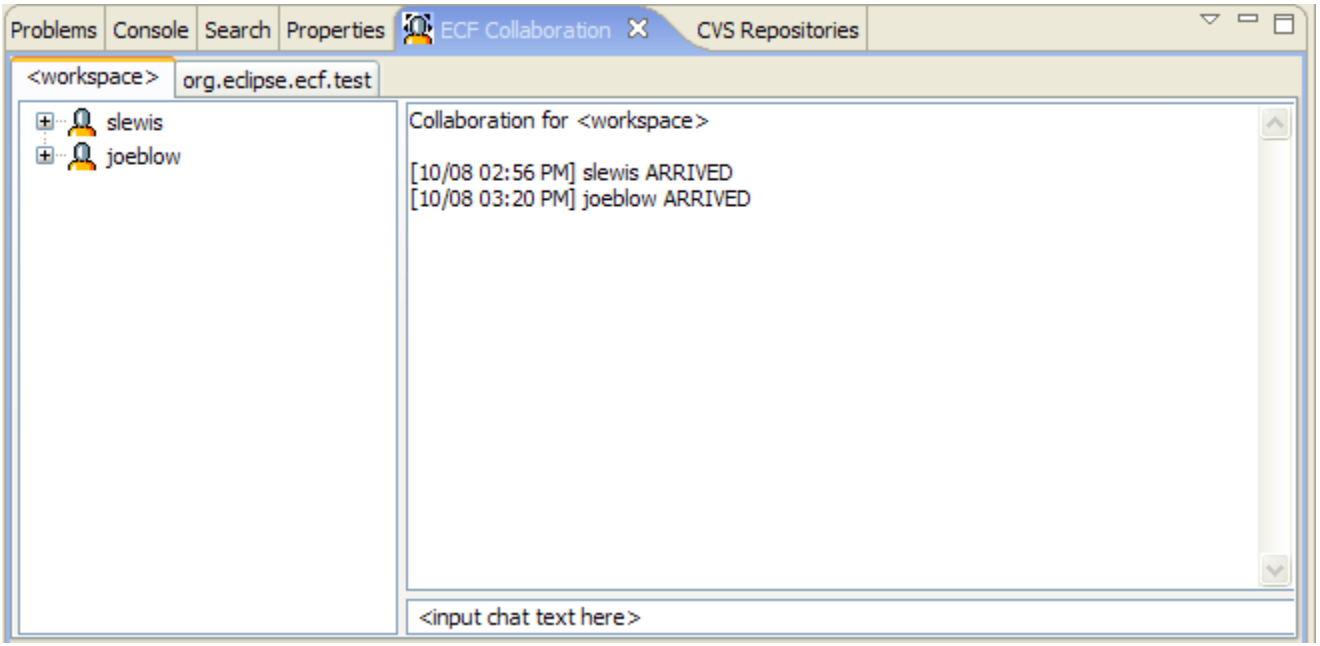
Scott Lewis -- slewis@composent.com
Mustafa Isik -- codesurgeon@gmail.com
Roland Fru -- roland@bitsvalley.com

Running Now: ECF 'Generic' server at ecf.eclipse.org



Tomcat+servlet bridge+Equinox+ECF plugins

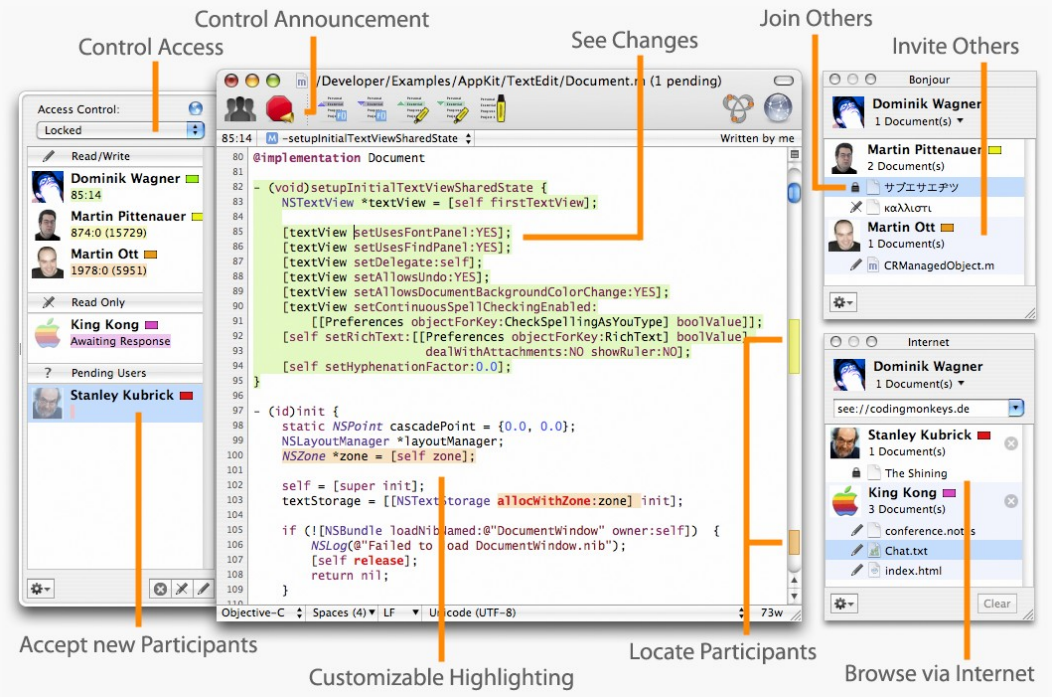
ECF Collaboration Example Application



Chat, URL sharing/co browsing, file transfer, remote control

Eclipse and RCP Client/Server and Peer-to-Peer Applications coming

- Subthaedit for Eclipse, EMF, editor X?



ECF + Bundles = Common Messaging APIs for Servers, RCP apps, Eclipse plugins

- High-level Messaging API bundles
  - Service Discovery
  - Presence and Chat
  - Data Channels
  - File Transfer
  - Remote (OSGi) Services
- + Multiple Transport bundles
  - Raw TCP
  - JMS/ActiveMQ
  - XMPP
  - JXTA
  - Zeroconf

Everything is bundle. Dynamically loaded/unloaded, developed/tested/debugged with PDE

ECF Remote Service API (new!)

- On the 'Service' OSGi process
  - Register remote service
    - IRemoteServiceContainer.registerService
  - Underlying service registry is **replicated** to all participating clients (using ECF provider)
  - Separate from OSGi service registry

ECF Remote Service API...on Client

- On the 'Client' OSGi Processes
  - `IRemoteServiceReference[] refs = getRemoteServiceReferences(...);`
  - `IRemoteService rservice = getRemoteService(refs[0]);`
  - THEN, 4 explicit options for remote service invocation
    - `1. Object proxy = IRemoteService.getProxy()`
      - Call/return. Blocks until result
    - `2. Object result = IRemoteService.callSynch(IRemoteCall)`
      - Call/return. Blocks until result
    - `3. IRemoteService.callAsynch(IRemoteCall, listener)`
      - Call/return. No blocking (listener notified)
    - `4. IRemoteService.fireAsynch(IRemoteCall)`
      - 'Fire and go'. No block (no success/failure info)

ECF Remote Service API: Summary

- Looks very similar to OSGi services
  - BundleContext.registerService
  - BundleContext.getServiceReferences(...)
  - BundleContext.getService(ref)
- Remote Registry Separate from OSGi Registry (not transparent)
  - Remote services explicitly identified as such (by registerRemoteService)
  - Clients may invoke services synchronously or asynchronously...choice based upon desired runtime behavior, timing, need for failure info
  - e.g. Axis 2
- Transport can be: JMS, ECF generic, XMPP, RMI, XML-RPC, SOAP, others

API Introduction

- Interoperability through protocol
  - `org.eclipse.ecf.core.IContainer`
  - Goal
    - Simple API / Extensibility via OSGi model / `getAdapter(…)`
- Clients use the `IContainer` API
  - `IContainer container = ContainerFactory.getDefault().createContainer("ecf.xmpp.smack");`
  - `Container.connect(…)`
- Semantics
  - Connection/Disconnection/LifeCycle

```
c.connect(ID, IConnectContext)
…
c.disconnect()
```

  - Protocol Adapters – getAdapter(…) abuse…
    - container.getAdapter(<interface>);

```
IFileshareContainer fsc = (IFileshareContainer) c.getAdapter(IFileshareContainer.class)
```

API Introduction

- `IAdaptable` abuse (we love the adapter pattern)
  - Presence/IM/Chat
  - Dynamic Service Discovery (zeroconf, etc…)
  - Datashare (channels)
  - File sharing
  - Call (SIP…)

API Introduction

- Two Extension Points
  - `org.eclipse.ecf.containerFactory`
    - ECF providers can implement their own `IContainer`
      - Current
        - XMPP/Jabber, IRC, JMS, Yahoo
      - Future
        - SIP, JXTA, Jingle, Sametime, AIM, etc…
  - `org.eclipse.ecf.namespace`
    - ECF providers can implement their own addressing
      - e.g., xmpp://zx@ecf.eclipse.org

Demos

Conclusion

- Support from the community **welcomed** and **appreciated**!
  - We need committed committers!
- Website
  - http://www.eclipse.org/ecf
- Mailing List
  - http://dev.eclipse.org/mailman/listinfo/ecf-dev
- Newsgroup
  - news://news.eclipse.org/eclipse.technology.ecf