

Eclipse Communications Framework

<http://www.eclipse.org/ecf>

John Beatty

Ken Gilmer

Scott Lewis

Pete Mackie

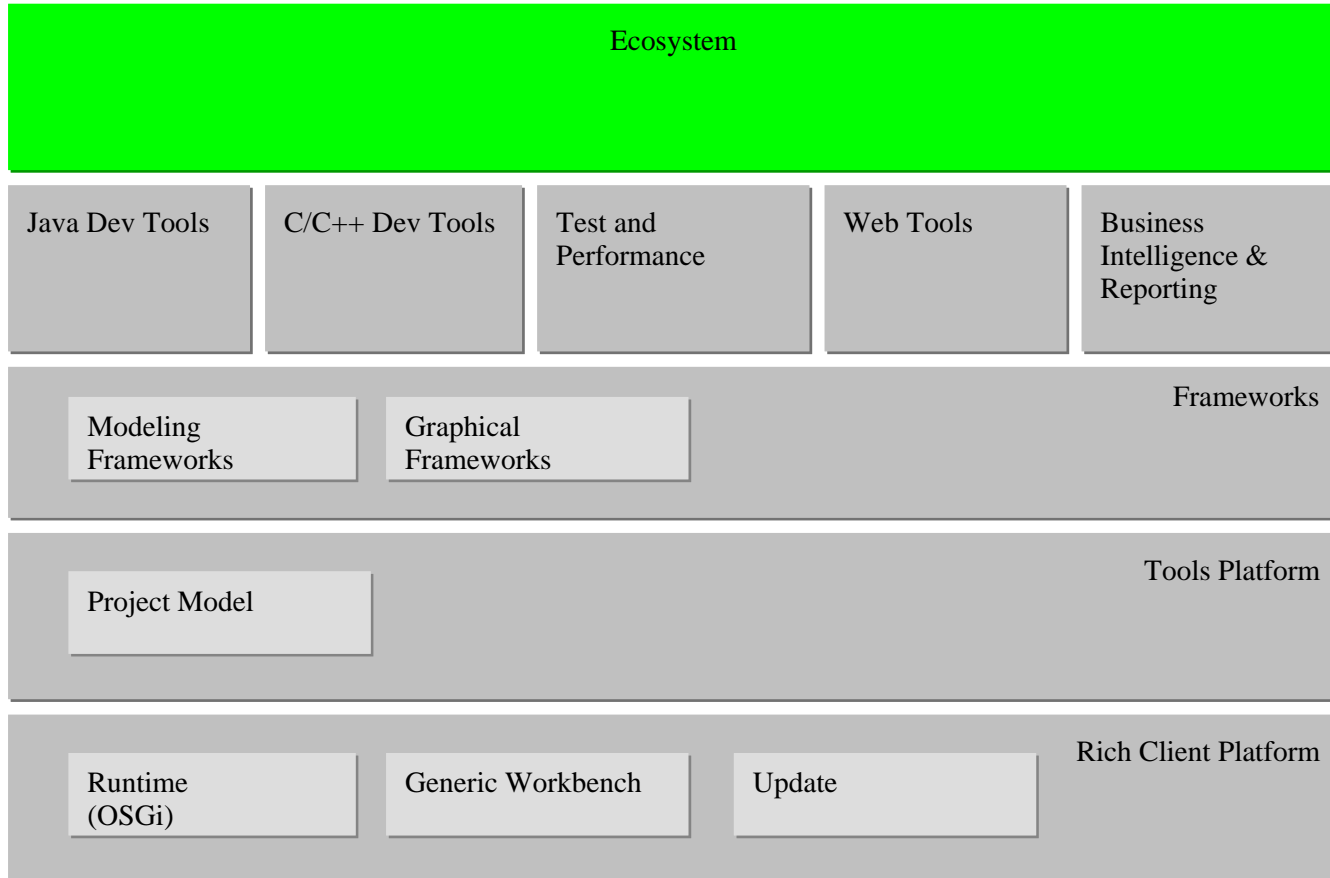
Peter Nehrer

Mary Ruddy

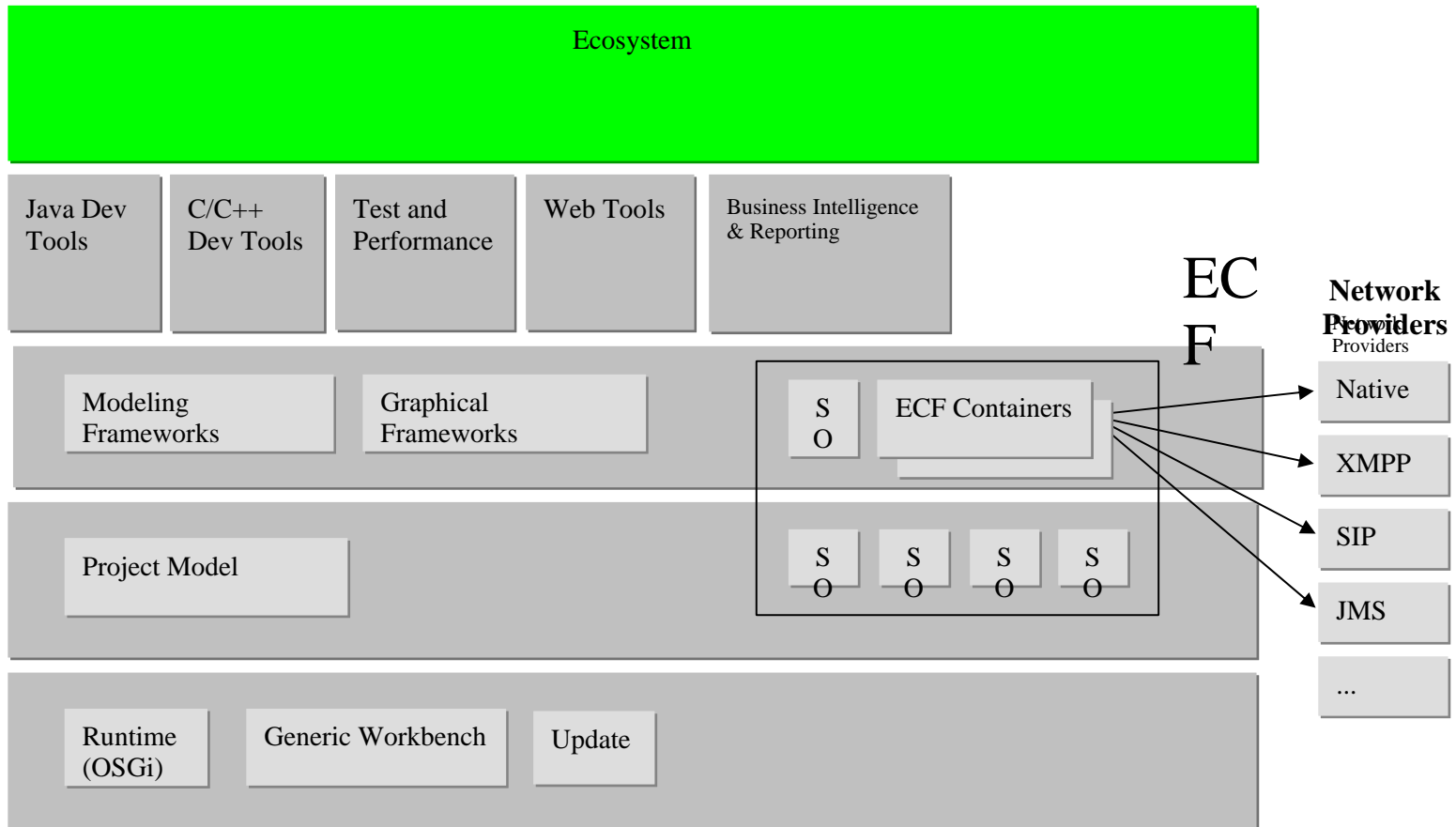
Rhett Savage

Paul Trevithick

Eclipse Architecture



ECF Containers



Why Another Framework/Abstraction Layer?

- Answer: **Communications Interoperability**
- Goals
 - Isolate protocols from component and/or application code
 - Introduce 'lightweight container'
 - No client-server assumption...assume only dynamically changing 'group of peers'
 - Peers **all** potentially contribute state (and code) to distributed app – via **shared objects**
 - Consistent session, messaging, and reliability semantics
 - Eclipse-based clients that use multiple-protocols-per-app
 - Try to make (reliable) distributed applications development **much easier**

How: A higher-level comm API

- Simple/Minimal/Small (< 100K for core interfaces/classes)
 - Join/Leave/Group membership info...**that's it!**
- Open Implementations
 - Anyone can implement a provider as extension
 - Existing protocols can be implemented open source or not
- Runtime Extensible
 - Adapters

Container-Provided Services

- Group Join/Leave
 - `joinGroup()`, `leaveGroup()`
- Group Membership/Failure Detection
 - `getGroupMemberIDs()`, `getGroupID()`
 - Critical for reliability
- Lifecycle and context for SharedObjects
 - `createSharedObject()`, `addSharedObject()`,
`removeSharedObject()`
- Event Notification
 - `addListener/removeListener`
 - Asynch events: `ISharedObject.handleEvent(Event)`

Example Client Code

```
// Create container instance via named provider
ISharedObjectContainer container =
SharedObjectContainerFactory.makeSharedObjectContainer("org.eclipse.ecf.provider.generic.Client");

// Create service id
ID groupID =
IDFactory.makeStringID("ecftcp://comosent.com:3282/server");

// Connect
container.joinGroup(groupID,null);

// Create/add ISharedObject
ISharedObject sharedObject = new HelloSharedObject();

container.getSharedObjectManager().addSharedObject(IDFactory.makeGUID(), sharedObject, new Properties(),null);
// shared object component communicates here

// Disconnect/Dispose
container.dispose(0);
```

Shared Object

```
public class HelloSharedObject implements ISharedObject {  
  
    ISharedObjectContext config = null;  
  
    public void init(ISharedObjectContext config) throws  
        SharedObjectInitException {  
        this.config = config;  
    }  
  
    public void handleEvents(Event event) {  
  
        // Handle all container-provided events here!  
        System.out.println("got event: "+event);  
  
        // Send message to all in group  
        ISharedObjectContext context =  
            config.getContext().sendMessage(null,"hello world");  
    }  
  
    public void dispose(ID container) { config = null; }  
}
```

ECF Providers

- **Implement ECF core Extension Points: containerFactory, namespace**
- **Implemented so far: ECF Generic, XMPP/Jabber**
- **Planned: SIP, JMS, JXTA, JGroups, Bitorrent, ...**
- **Can support both open and proprietary network protocols (legacy and new)**
- **Appeal**
 - **1st choice: Consider implementing your favorite provider and contributing to ECF**
 - **2nd choice: Ask us to do it for you**

Provider Implementation

```
// In your plugin.xml
<extension
    point="org.eclipse.ecf.containerFactory">
    <containerFactory
        class="com.myco.MyContainerInstantiator" name="generic">
    </containerFactory>
</extension>

public class MyContainerInstantiator
    implements ISharedObjectContainerInstantiator {

    public ISharedObjectContainer
        makeInstance(SharedObjectContainerDescription description,
                    Class[] argTypes,
                    Object[] args)
            throws SharedObjectContainerInstantiationException {

        // Make new container instance here of appropriate type
        return new FrobozSharedObjectContainer();
    }
}

SharedObjectContainerFactory.makeSharedObjectContainer("generic")
```

Applications: Human-Human Communication

- **File Sharing**
- **IM/Chat**
- **App Sharing**
- **VOIP**
- **Conferencing**
- **Drawing/Whiteboard**
- **Multi-Player Games**
- **??**

Tool Communication (plugin-to-plugin)

- **GraphShare: Shared Model Editing**
- **Shared Debuggers/Simulation**
- **Workflow**
- **Collaborative Content Creation/Mgmt**
- **??**

Existing Codebase: Core Plugins

- **org.eclipse.ecf**
 - Framework interfaces, factories, and exception classes
- **org.eclipse.ecf.provider**
 - ECF provider interfaces and classes
- Runtime Requirements: JRE 1.4+, OSGI 3.0 **only**

Other Infrastructure Plugins

- **org.eclipse.ecf.sdo**
 - Graph share framework – talk about this shortly
- **org.eclipse.ecf.ui**
 - Communications UI code: e.g. Abstract 'Buddy List View'
- **org.eclipse.ecf.doc**
 - Eclipse help docs: API docs, overview docs, etc

Example App Plugins

- **org.eclipse.ecf.example.collab**
 - Real-time collaboration application (url sharing, file sharing, chat, etc)
- **org.eclipse.ecf.example.sdo.editor**
 - Graph share example application
- **Coming (very) soon**
 - Jabber Provider: `org.eclipse.ecf.provider.xmpp`
 - Apps
 - Shared Editor for Team Outlining (SETO)
 - (Distributed) Team Project Planning Tool

What is GraphShare?

- Replicating graph data structures
- SDO Data Graphs (using EMF ref impl)
- Serialized Form is XML
- Participants distribute and then receive updates, apply changes to local replication
- Framework supports extension/customization of
 - Replication algorithm – when and where
 - Change propagation -- when
 - Conflict resolution – how to resolve
 - Adding extension points to org.eclipse.ecf.sdo right now

Default Graph Share Implementation

- Updates optimistically – any participant may commit at any time
- Updates must be received in sequence
 - Otherwise, all subsequent updates to the replica fail
 - Clients handle this (e.g., re-subscribe)
- New group members are initialized arbitrarily
 - All members send their Data Graph copy
 - The first one received is accepted
- No server/coordinator required

Default Update Provider

- EMFUpdateProvider
 - Uses EMF-SDO API to
 - Extract local changes
 - Serialize
 - Deserialize on other end
 - Apply to remote replicas
 - Reverses local changes when applying remote updates (this may be configurable in the future)

Example: Shared Model Editing

- Extended SDOEditor
- Supports editing of arbitrary EMF models with SDO support
- When saving, broadcasts changes to group members
- Remote updates immediately saved
- Shared models are identified by their workspace-relative path

Future

- Respond to Community Feedback
- More Providers: JMS, JXTA, Jgroups, SIP, **your** favorite
- More Applications: GEF Graph Share, VOIP, **your** favorite
- Integration with RCP apps
- Integration with Eclipse Trust Framework
- More/Better Libraries for Protocol Patterns
 - e.g. Both optimistic and pessimistic protocols...e.g. transactions
 - Replicated Data Structures (e.g distributed hashtable)
- Layered APIs for user presence, other comm/collab componentry
- Dynamic Lookup and Discovery (?)
- Others Requests (?)

Consider Participating!!

Feedback from you on Desired Directions

Build Communications Components (shared objects) and Apps

– License them if you want to!

Implement Containers/Providers

Contact: <http://www.eclipse.org/ecf> newsgroup, OR

slewis@composent.com, pete@sequest.com

Special Thanks

Eclipse Foundation: Mike M, Bjorn FB, Brian B, John W

OTBC: Open Technology Business Center

IBM Jazz Team: Li-Te Cheng, John Patterson

All Team Members: They are doing on own dime/time

Thanks to All Families