Eclipse Device Debuging:
Debugger Services Framework (DSF)

Martin Oberhuber and Pawel Piech, Wind River

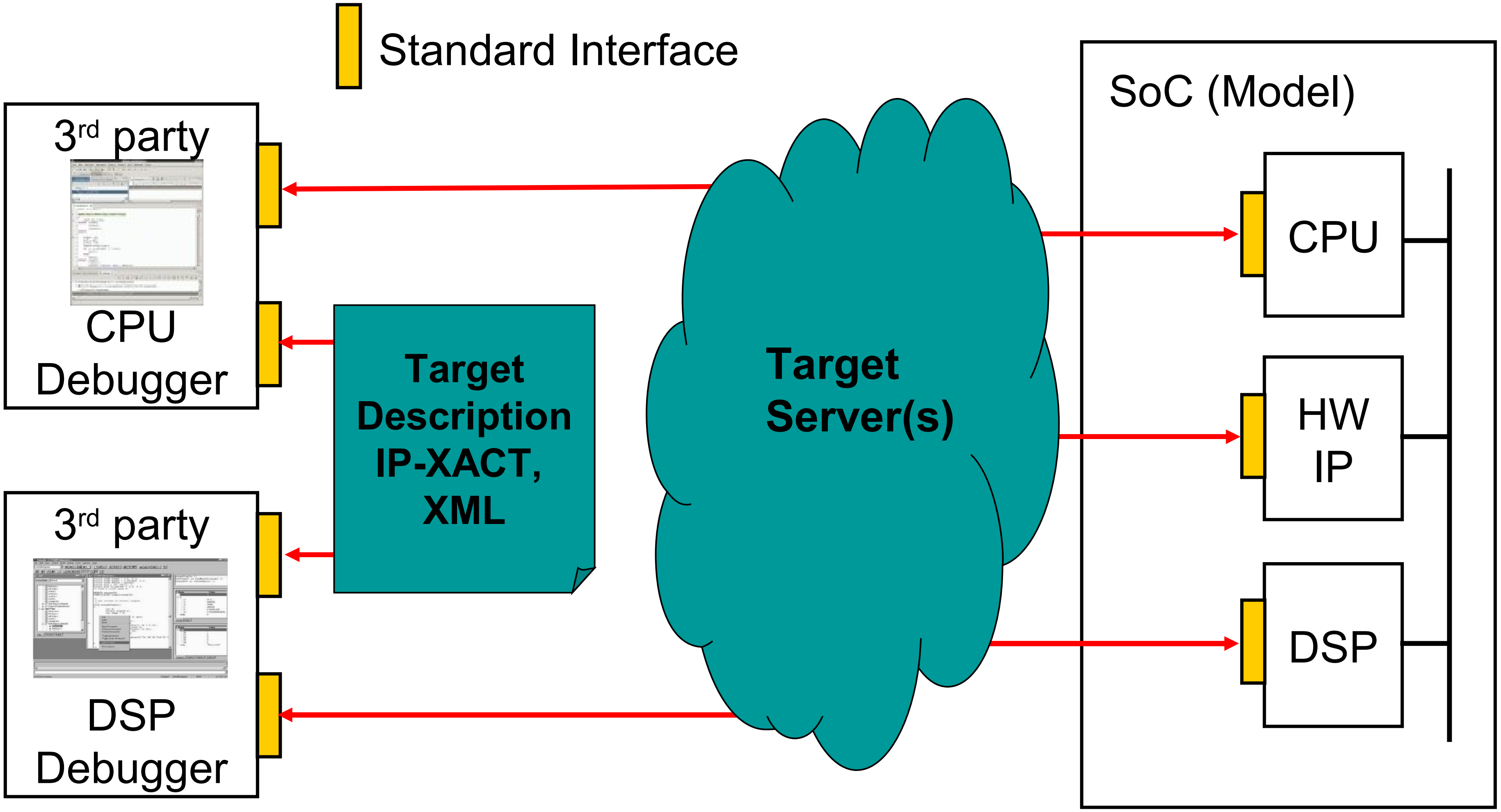
ECSI Workshop on System Debug, 10-Mar-2008

Eclipse Device Debugging Project

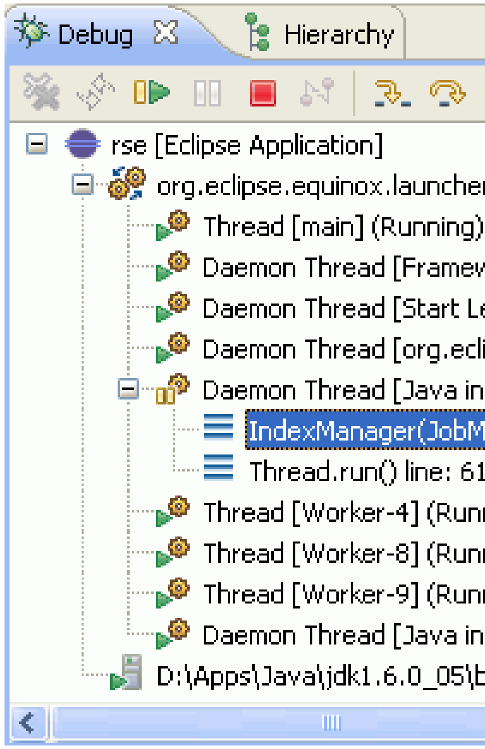
- Mission: ***Build enhanced debug models, API's, and views that augment the Eclipse Debug Platform in order to address the added complexities of device software debugging.***
- Wind River (lead), Ericsson, IBM, Mentor Graphics, Nokia, PalmSource, Symbian, TI, QNX, Freescale
- Initiatives
  - Debug Views – Flexible Hierarchy
  - Debugger Services Framework (DSF)
  - Memory View, Multi-Context, Disassembly
  - SPIRIT / IP-XACT Editor
  - Target Communication Framework (TCF) – on TM Project

Pawel Piech
DD Lead

System Debug: the Big Picture



History: Eclipse Platform/Debug


Debug ☒ Hierarchy

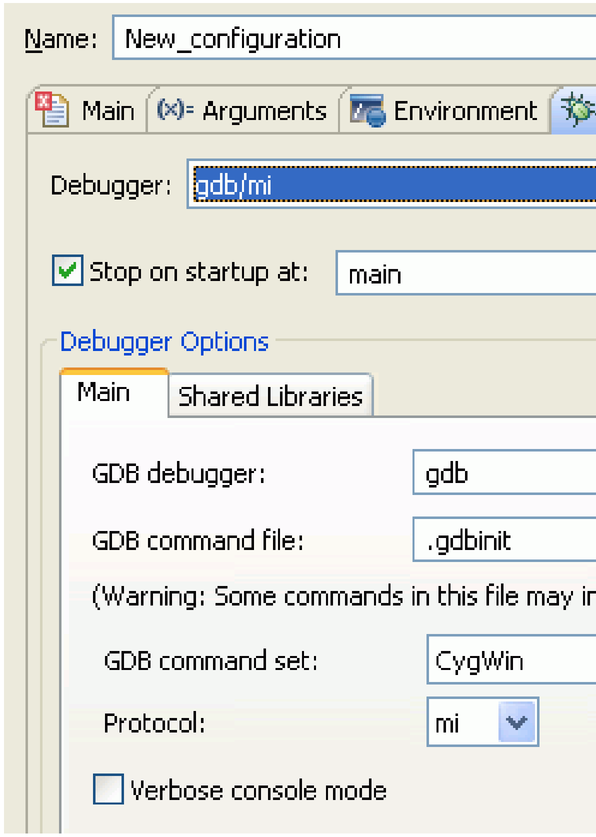
```
rse [Eclipse Application]
  org.eclipse.equinox.launche
    Thread [main] (Running)
    Daemon Thread [Framew
    Daemon Thread [Start Le
    Daemon Thread [org.ecli
    Daemon Thread [Java in
      IndexManager(JobM
      Thread.run() line: 61
    Thread [Worker-4] (Run
    Thread [Worker-8] (Runn
    Thread [Worker-9] (Runn
    Daemon Thread [Java in
  D:\Apps\Java\jdk1.6.0_05\b
```

- ILaunchConfiguration (*static*)
  - ILaunchConfigurationTab*
  - ISourceLocator, ISourcePathComputer
- ILaunch (*dynamic*)
  - IDebugTarget, IProcess, IThread, IStackFrame
- IBreakpoint (*static+dynamic, editor integration*)
  - IVariable, IWatchExpression

- Synchronous operation
  - Many Known implementations (JDT, CDT, PHPEclipse, PDT, RubyDT, DLTK, IMP, … WR Workbench 2.5 and before)

History: C Debug Interface (CDI)

- CDIDebugModel (*static*)
  - ICDIDebugger2, ICDebugConfiguration
- ICDISession (*dynamic*)
  - ICDIEvent*, ICDISignal*, ICDIRegister
- IAddress (*static+dynamic*)
  - ICDILocation, ICDIBreakpoint, …


Name: New_configuration

Debugger: gdb/mi

[✓] Stop on startup at: main

GDB debugger: gdb

GDB command file: .gdbinit

(Warning: Some commands in this file may i

GDB command set: CygWin

Protocol: mi

[ ] Verbose console mode

- Synchronous operation
  - Many Known implementations (CDT and derivatives, e.g. Nokia Carbide, ARM, … but **not** WR Workbench)

Some Problems of Existing Approaches

- Fixed Hierarchy
  - ILauch – IDebugTarget – IProcess – IThread – IStackFrame
  - But how to map **mulitple Cores** on a Debug Target?
- Problematic Integration of Multiple Debug Engines
  - Monolithic – Hard to do 3rd party value-add (→ TCF!)
  - Mixed stack view e.g. Java – JNI – Native; breakpoints
  - Compare data from 2 debuggers in a variable view
- Synchronous Operation
  - To evaluate a stack, variable… start a Job (which just wait on the underlying debugger's response most of the time)
  - Scalability, Synchronization problems Jobs – Model – View
- Fixed Update Policies
  - One Debug Event – All Debug Views updated: don't scale

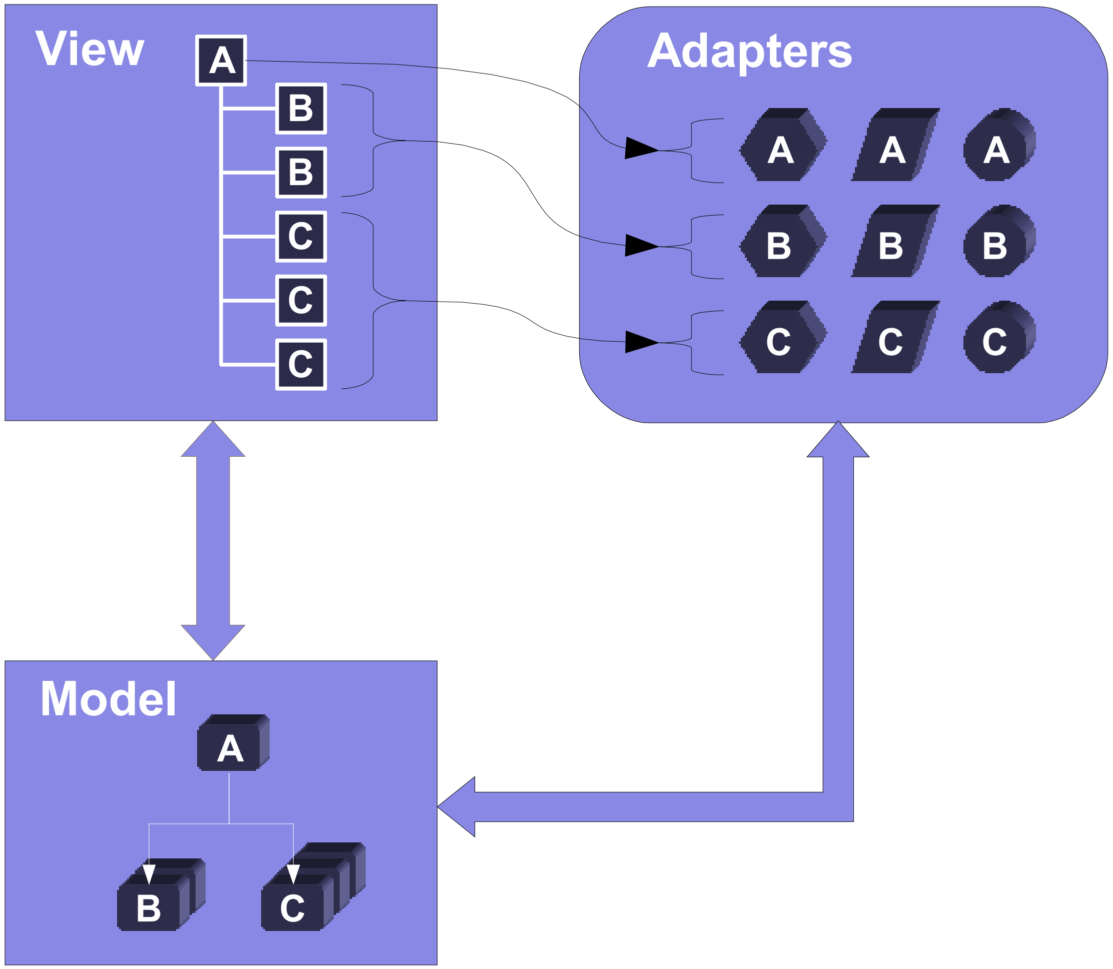
Wind River's History

- Before Eclipse: Multiple Debug Technologies
  - Tornado/gdb, Look!, SingleStep, VisionClick, Diab RTA
- Unified Proprietary Technology under Eclipse
  - Back-End: on dfwserver (mostly based on SingleStep)
  - Debug Model: Riverbed (mostly based on Diab RTA)
  - Front end: Eclipse Platform/Debug
- Started Open Source Initiatives
  - Device Debugging (2005) – Goal: Improve Platform/Debug
    - Strong vendor participation (almost everyone including IBM)
    - First Results: Debug Flexible Hierarchy, Memory Renderings
    - More Initiatives: IP-XACT / SPIRIT, DSF, Disassembly
  - DSF (2006) – Riverbed to Open Source
  - DSF gdb/mi Reference (2007) – Ericsson and WR

Platform Flexible Debug Model (3.2+)

- First appeared as provisional API in Eclipse 3.2
  - Main architect Darin Wright (IBM) based on DD discussions
  - Refactored to use JFace Viewer for Eclipse 3.3
  - Most APIs changed in 3.3 – will still be provisional in 3.4
  - Reference: EclipseCon presentations
- Customization of standard debugger views (Debug, Variables, Registers) look and feel
- API for populating these views with minimal assumptions about structure and format of data
- Pluggable Update Policies

Debugger Views - Flexible Hierarchy

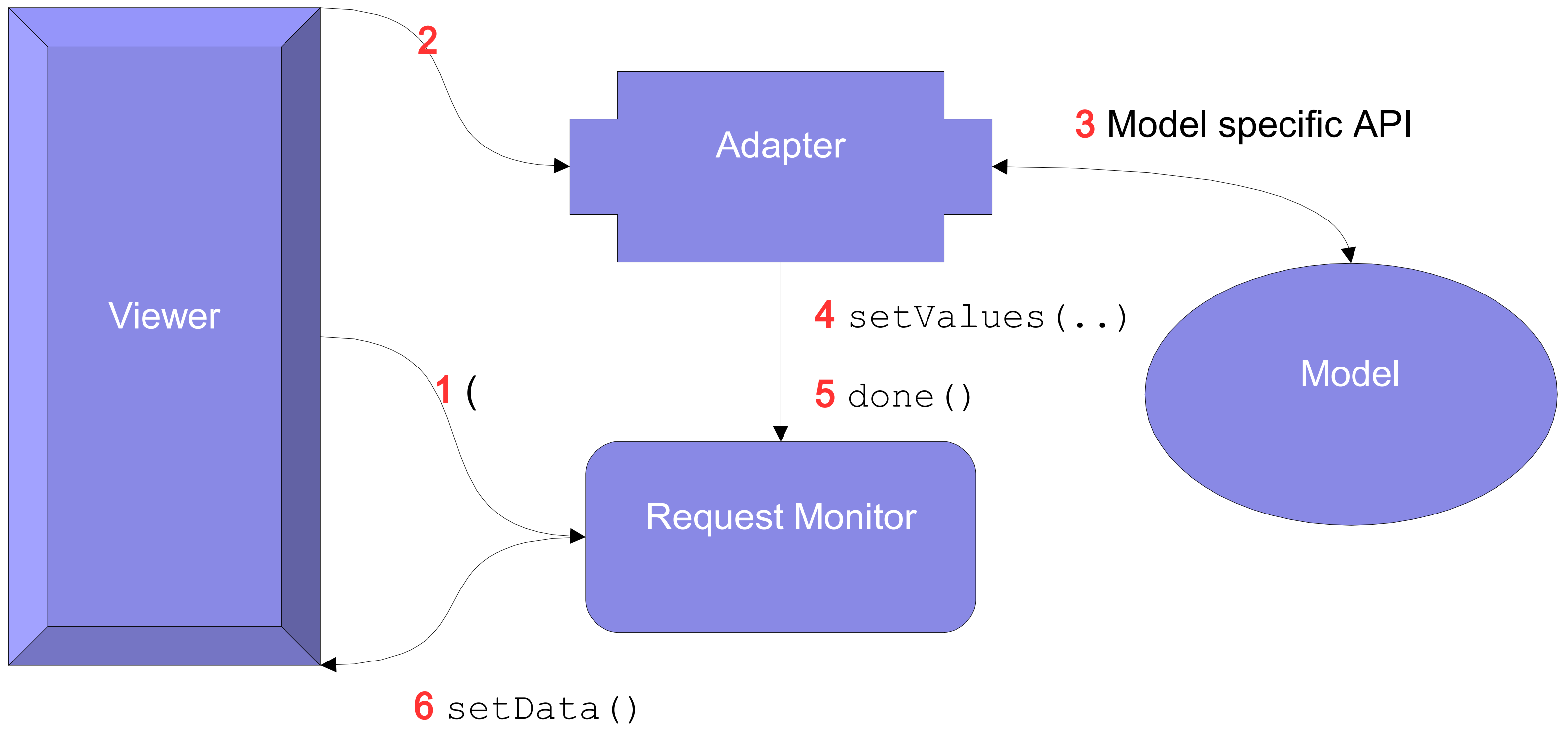


Debugger Views - Flexible Hierarchy

- Adapter Types – Each adapter provides a property for elements:
  - `IElementContentProvider` – children
  - `IElementLabelProvider` – text, icon, font, color for each column for an element
  - `IModelProxy` – model event handler, translates events into view update requests
  - `IColumnPresentation` – list of columns
  - `IElementEditor` – a modifier and cell editors for each column
  - `IElementMementoProvider` – seralizable data
  - `IViewerInputProvider` – proxy input into a viewer

- Eclipse 3.2+ comes with **predefined adapters** to mimic the old Platform/Debug behavior, but uses Flexible Hierarchy internally.

Debugger Views - Flexible Hierarchy



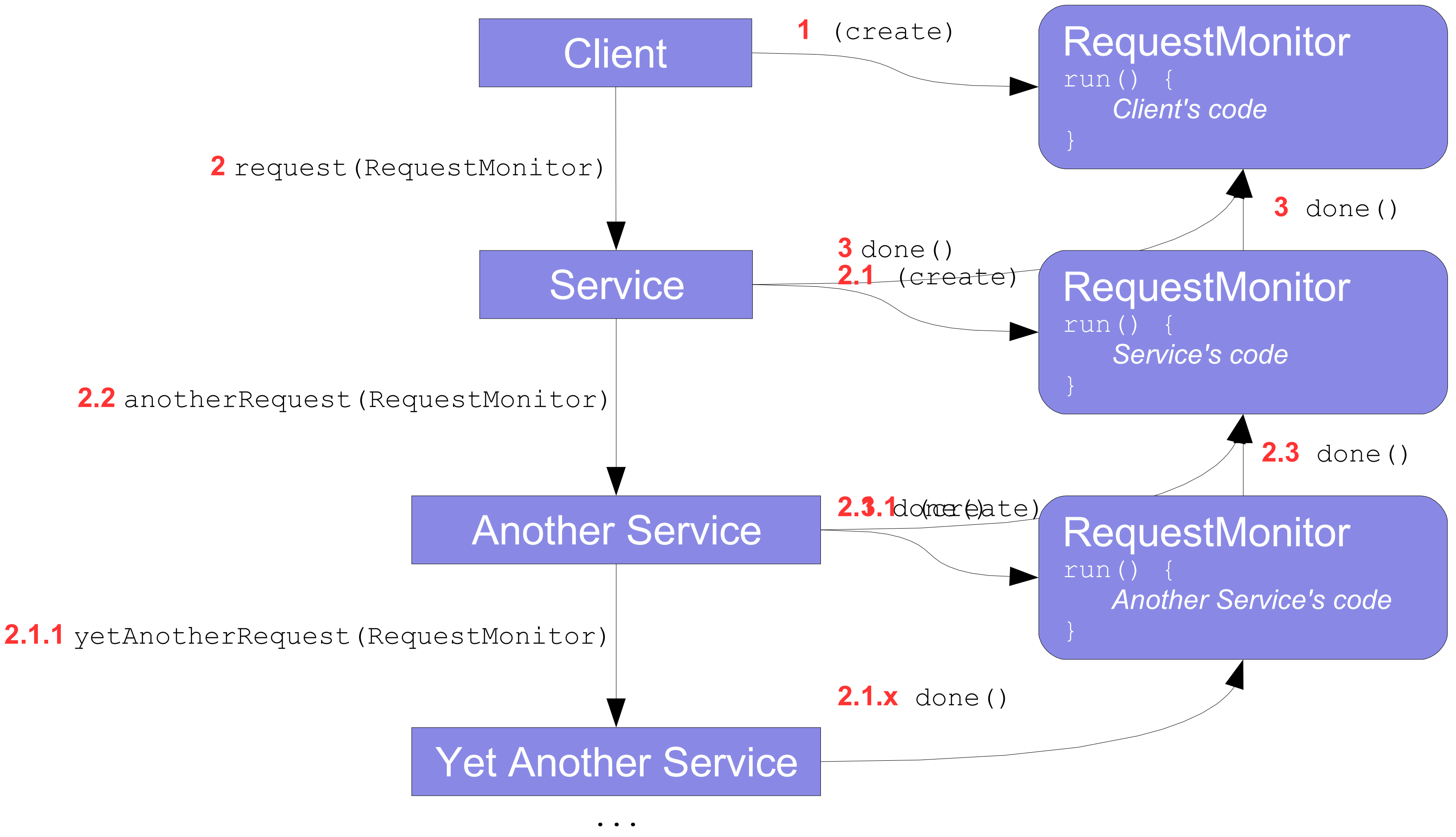
DSF (Debugger Services Framework)

- A Layer on top of Flexible Hierarchy to simplify its use
- API to accommodate needs of embedded debuggers: **performance, modularity, extendibility**.
  - DSF is based on Riverbed concepts but a Community Effort
- Part of DD project but trying to push into Platform
  - DSF 0.9 with Eclipse 3.3, running for 1.0 this year
  - Current WR Workbench 3.0 switched from Riverbed to DSF
- Dependencies
  - Java 1.5 (for util.concurrent: Executor)
  - CDT (for IAdress interface: to move into Platform)

DSF – Concurrency Model

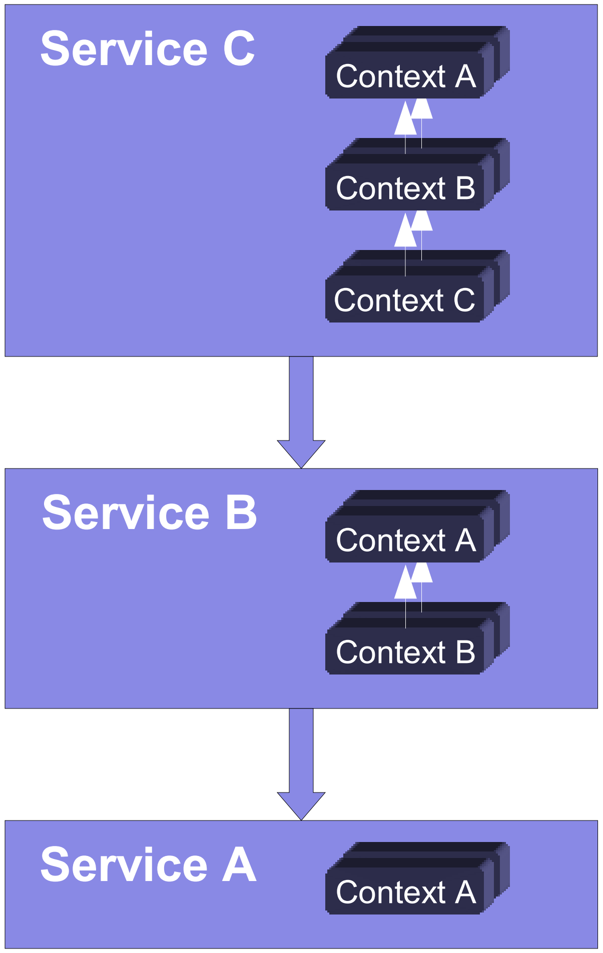
- All public APIs accessed on a single “session” thread
  - Managed by a Java 1.5 executor object
  - Thread-safe: session thread as a global lock for state accessible through public APIs of all the services
  - Services are still free to create separate worker threads to execute long-running operations
  - Same model as SWT and most other window toolkits
- Leads to an asynchronous request – callback model for most of the clients: better scalability and performance if many threads, operations, events

DSF – Asynchronous Interfaces



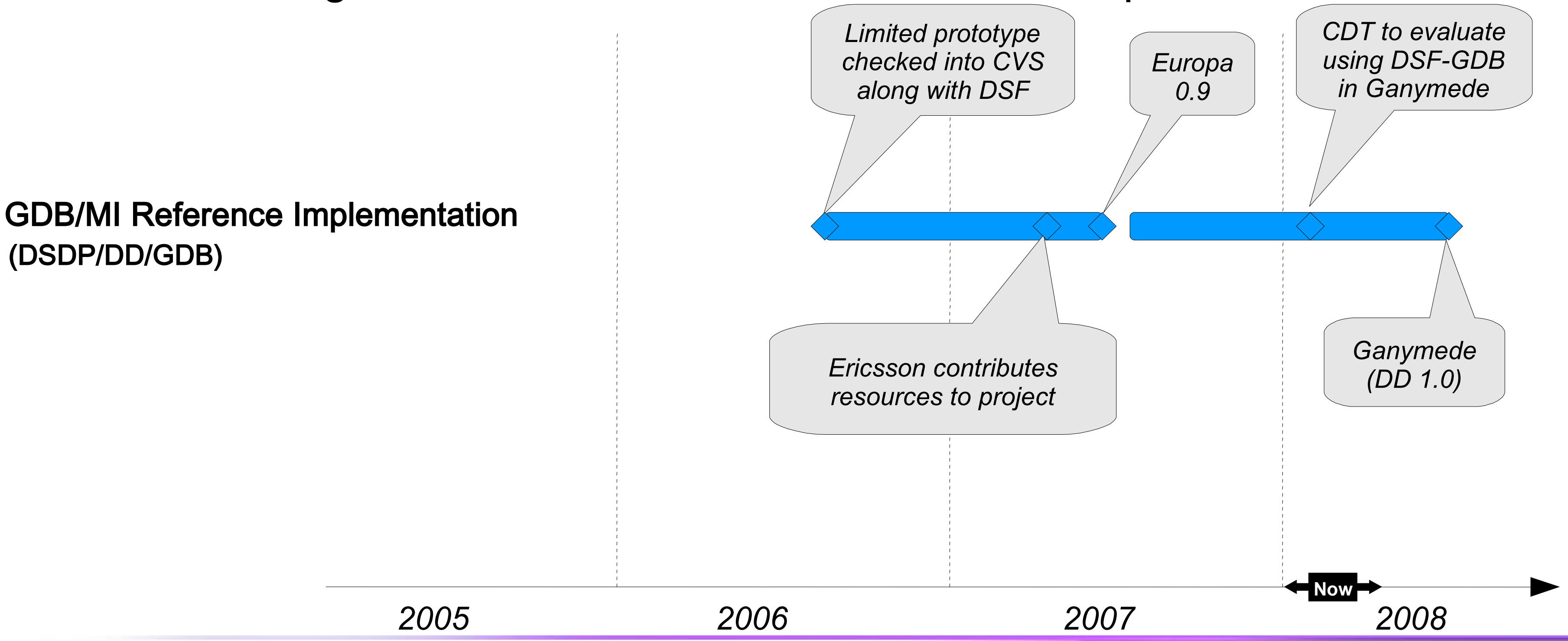
DSF – Data Model

- Services' data handles implement `IDMContext` interface
- Contexts are <u>immutable</u>, light-weight, and must properly implement `equals()` and `hashCode()`.
- A service can build on another service's context object to provide additional data
- Contexts are equal if all the contexts that they build on are equal
- Services accept generic contexts as arguments and search the context hierarchy for the relevant handle to act upon



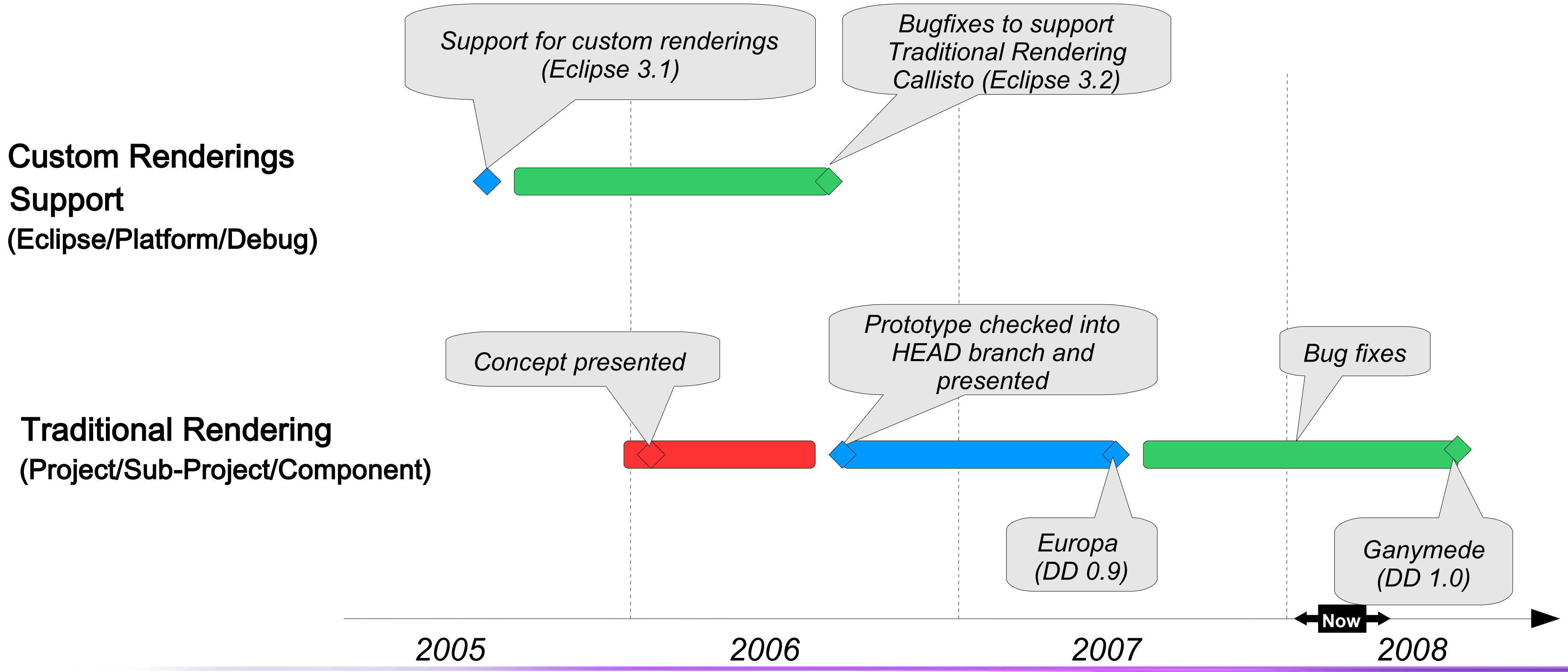
GDB/MI Reference Implementation

- Create a GDB-based debugger which implements DSF model APIs (functionally equivalent to the GDB debugger using CDI and standard debug model)
- Tuned for gdb 6.7; to remain in DSDP-DD for Eclipse 3.4



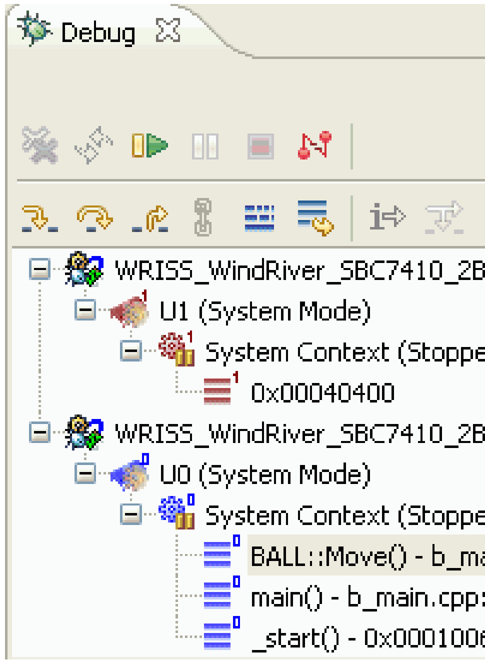
Other DD Initiatives: Memory View

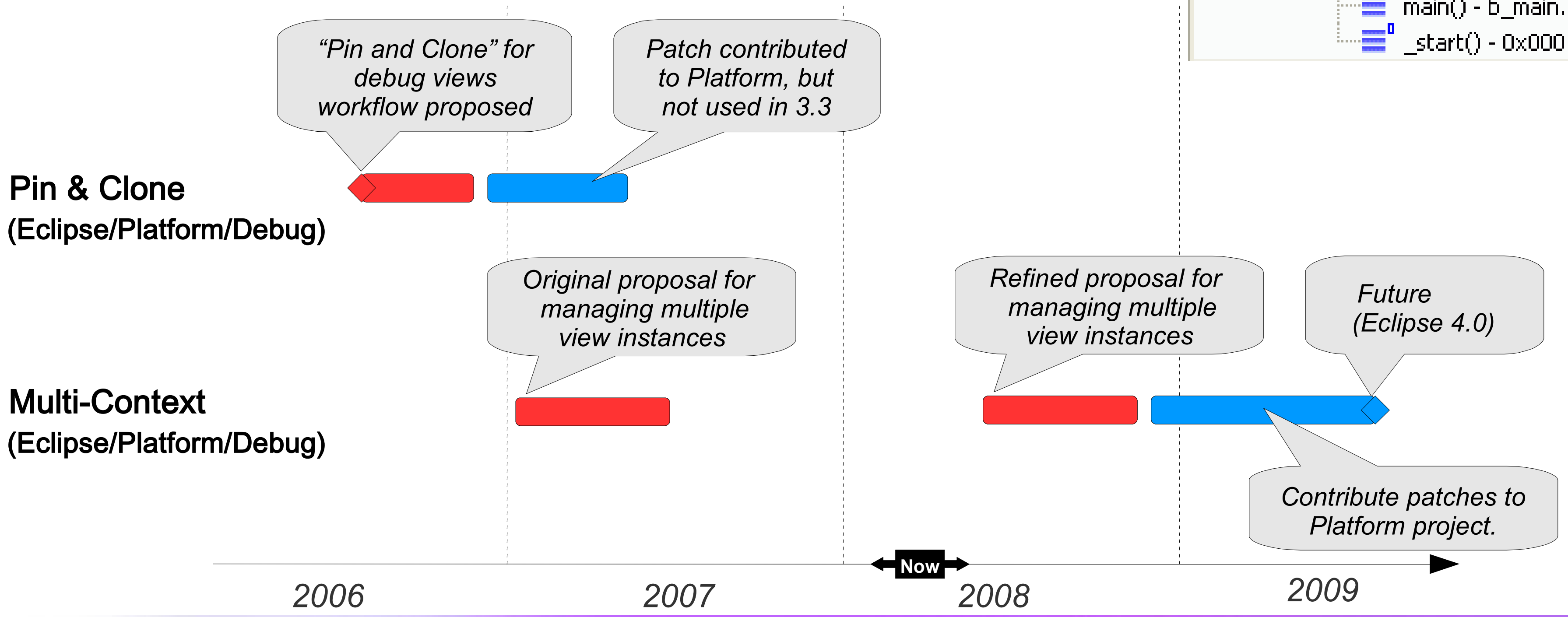
- Provide memory view support suitable for Embedded development (pluggable Rendering Implementation)
- “Traditional Rendering” complete since Eclipse 3.3



Other DD Initiatives: Multi-Context

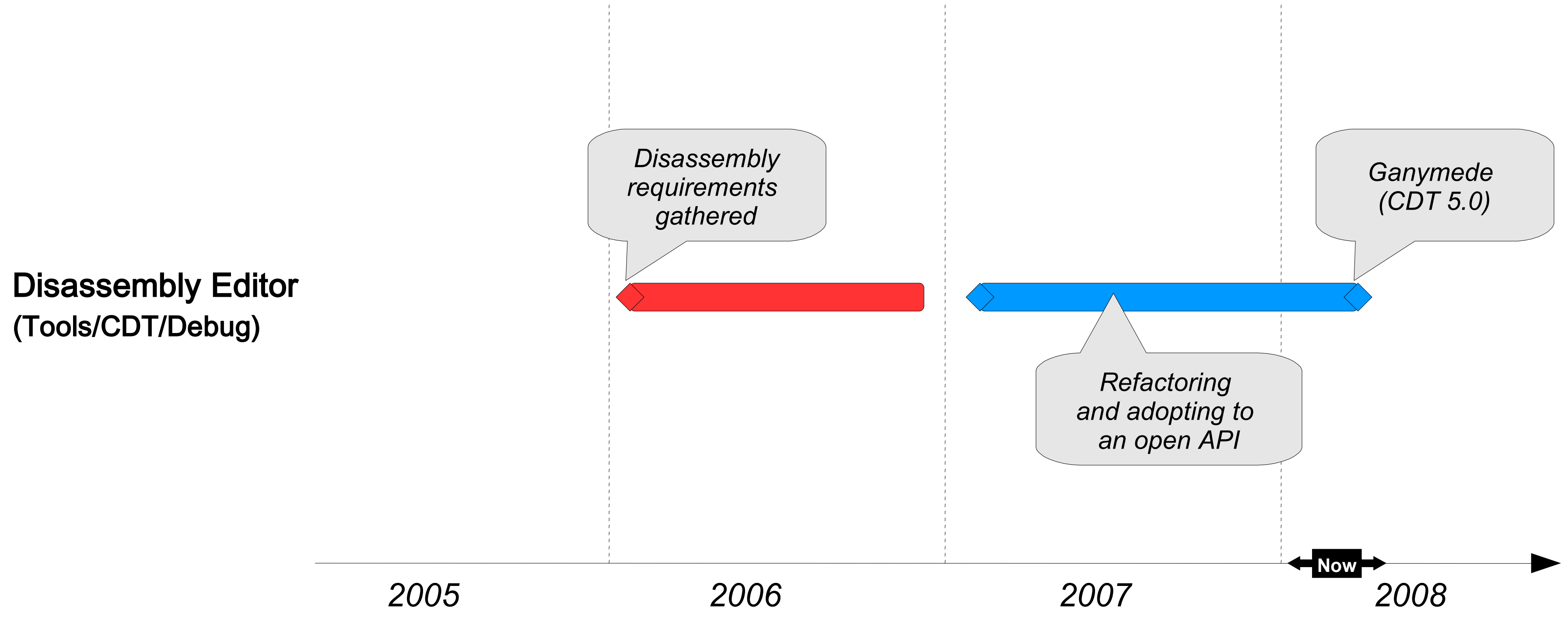
- To improve workflows and context switching when debugging multiple threads, processes, targets, etc.
- "Colored Views" in WR Workbench
- Proposed patches to Platform but likely not in 3.4




"Pin and Clone" for debug views workflow proposed
Patch contributed to Platform, but not used in 3.3
Pin & Clone (Eclipse/Platform/Debug)
Original proposal for managing multiple view instances
Refined proposal for managing multiple view instances
Future (Eclipse 4.0)
Multi-Context (Eclipse/Platform/Debug)
Contribute patches to Platform project.
Now
2006
2007
2008
2009

Other DD Initiatives: Disassembly

- To provide a disassembly editor and replace existing CDT disassembly view.
- In Progress at ARM but likely not complete for 3.4



References

- Eclipse Platform/Debug
  - http://help.eclipse.org/help33/
- Flexible Hierarchy
  - EclipseCon tutuorial presentations 2006, 2007, 2008
- DSF Architecture Docs
  - http://dsdp.eclipse.org/help/latest
  - EclipseCon tutorial presentation 2008
- Device Debugging Overview
  - http://www.eclipse.org/dsdp/dd/

Questions?