DSDP - Device Debug Subproject Meeting
19 July – 20 July 2005

Meeting Details

Time: Tuesday July 19th (noon) to Wednesday July 20th (4 pm)

Location: IBM Toronto Software Development Lab
(http://www.can.ibm.com/torontolab/)
8200 Warden Ave. East
Markham, On
L6G 1C7

Meeting Room: Willow Education Room
Conference number: 1-866-576-2504 conference id: 6392948.

Attendees

IBM
Alan Boxall
Samantha Chan
Pete Nicholls
Jeff T
Doug Schaefer (phone)
Darin Wright (phone)

AMI Semiconductor
Ken Dyck
Mark Melvin

Palm Source
Ewa Matejska

QNX
Alain Magloire
Mikhail Khodjaiants

Nokia
Warren Paul
Ken Ryall

Freescale
Daymon Rogers
John Cortell

Intel
Susan Macchia (phone)

Mentor Graphics
Mark Bozeman

HP
Sumit Sarkar was unable to attend

Wind River
Doug Gaff
Felix Burton
Randy Rohrbach
Ted Williams

Texas Instruments
Chris Recoskie
Andy Waterson
Brian Cruikshank
Dobrin Alexiev
Paul Gingrich
Martin Imrisek

Agenda and Notes

Introductions

Summarize current status of DSDP proposal

- DSDP is an official project
- Mailing lists, etc coming soon

Meeting Goals

- Pete really wants to get to the prototype stage after this meeting
- Next meeting will be a review of prototypes.

Nokia / Freescale goals

- Works on debugger at Nokia
- Ken Ryall – Nokia
- Warren Paul – Nokia
- Daymon Rogers – Freescale
- John Cortell – Freescale
- Nokia = Code Warrior for Symbian
- Technical goals
    - Reuse CodeWarrior debugger core (separate process)
    - Integrated to CDT via a CORBA API

  - Extend and enhance CDT debugger
  - Integrates on the CDI level
- Proposed features
  - Debugging of project-less executables
  - Enhanced variable formatting
  - C++ runtime type inspection
  - Enhanced breakpoints (actions, templates)
  - System Browser
  - Support for address spaces
  - Module symbol loading enhancements
  - Memory window enhancements
  - Display variable addressed or containing register in Variables view
  - Selectable scope for Variables view
  - Multiple thread debug context
  - Generic command line interface
  - Change the Program counter
  - Auto-update views
  - Cached-source for debugging
  - Linetable tick marks
- They connect directly to CDI
- Freescale is working closely with Nokia, but they haven't made any plans to ship their own Eclipse-based product.

AMI
- Assembly code development for DSP's
- Debugger written in Java to go with Eclipse
- Not using CDT
- Seeing similar issues as other folks
- Talking to embedded hardware over serial connections

AI: CDT, Parallel Tools, Eclipse Core, DSDP – all are getting debugger items. We should come up with an agreement on where to post debugger stuff. Maybe the solution is to copy Greg Watson.

Palm Source
- Want multiple-processes debug
- Using CDT and GDB

Debug core (org.eclipse.debug)
- Flexible hierarchy (Current debug core enforces a fixed hierarchy)
- Propose solution to allow for flexible parent/child relationships
(presenter: Alan Boxall)
- Solution 1: Use extension point to register hierarchy. Debug core would read and use this hierarchy.
- Solution 2: Use getAdapter code to "discover" hierarchy
- This would influence the content of views... e.g. Debug and Register

views
- Multiple language support
- Allow different "Targets" to exist under a single target

- See Alan Boxall's presentation - "Flexible Hierarchy" design proposal
- Goals: Allow debug plug-ins to define their own debug element hierarchy and behavior. Debug views become generic and can be customized.
- Hierarchy
    - o All objects implement IDebugElement
    - o Debug objects optionally implement IStep and ITerminate, but custom actions can also be added
    - o When debug session starts, it registers its "root" IDebugElement with the debug core
    - o Hierarchy is retrieved by getting the children of a debug element, and core makes no assumptions on hierarchy.
    - o Suggestions for improvement: getChildren() should be able to get a range and should be able to response asynchronously in case the process of determining children is blocking. The range should allow virtual updating (only update what you see and update as you scroll).
    - o How much should the model know about the views (for things like partial updating, update policy)? Darin think they should be tightly decoupled. Pete feels there should be some view knowledge is the model so the model can provide a better answer when the view asks for information.
- Generic debug view
    - o Core supplies base views for consistency, e.g. debug, variables, registers
    - o Providers can define their own views based on this or override core views
    - o Example: after sending the debug event, the view updater would also send a selection event.
    - o Group would like to have multiple debug views with clone/pin capabilities.
- Note that the Debug session is just the root of your hierarchy; it's not necessarily correlated with a single launch. This still needs to be worked out.

CDT and DSDP – Alain Magloire
- See Alain's presentation "CDT Debug Overview", especially the new hierarchy
- CDT Debug Core Interface – will break into 2 blocks
    - o CDT Debug Core Interface
    - o CDT Abstraction Implementation
- CDI will be optional. You can instead override the CDT Abstraction Implementation.
- DSDP will modify the Eclipse Debug UI and Debug Core, and CDT will provide a default implementation of this for C development with GDB. CDT

will also be a sample implementation that can be copied and customized for the specific customer scenario.

- One of the most important things for CDT is to share common code.

Debug views
- View update policies (presenter: Samantha Chan)
- Propose that models drive view updates instead of current method of views reacting to a debug event
e.g in R31 after a step over the model is asked 27 times "Can you step over?" in attempt to get the correct enable/disable state of actions and buttons

- See Samantha Chan's presentation – "Update Policy"
- Requirements
  - o Make debug views update more flexible
  - o Allow a view to have different update policies
  - o Allow models to specify different policies for different views
- Proposal: Replace single View Updater with Pluggable View Updaters. Also, handle both selection events and debug events.
- Examples: selection event updater, debug event updater, delayed debug event updater, timer updater, preference store changed updater, update on demand only.
- Updates are responsible for controlling when a view gets content from the model.
- The "when" part of the update isn't needed by Java, since Java updates are strictly event-driven.
- The desire would be to have a set of default updaters that would cover more situations.

Debug Console

- *Ability to send commands to the debugger backed*
- *Since different debug engines are used it may be beneficial to have a set of common commands that maps to code debug API calls. Since everybody have different naming preferences we should consider allowing the names to be configurable*
- *Some possible requirements:*
  - o GDB command line, plus potentially some additional commands
  - o Scripting – would need a pluggable language
  - o This is debugger backend console
  - o AI: WR's serial/network console is one possibility for the view itself
  - o Eclipse 3.1 has a console view that is abstractable.
  - o One console view that is debug view selection-dependent.
  - o Could follow the key bindings model and have command bindings for the same set of actions.
- AI: Nokia will write a proposal and requirements.

Logical Structures – who is using?

- *Capability to specify what members of a structure are viewed and how*
- ExtensionPoint - LogicalStructureTypeDelegate
- Nokia investigated for customer data types.
- Java uses these extensively, also Java detail files.
- User-defined ones are the most interesting.
- In IBM's commercial product, they will provide a default set of these for their customers, but they don't expect customers to do that on their own.

Memory View

- Demo by Chris
- OOB comments:
    - Where to put address?
    - Change "memory monitor" to "memory location"
    - Targets with different widths, and monitors apply to specific targets.
    - Change highlighting – should be in red. Currently have to override the model to make it red.
    - Bug fixed: selecting a register blanks the memory view
    - Maybe add a hot key to hide memory block selection region
    - Need a quick way to add an address – maybe an edit box
    - Need a way to switch rendering for an existing memory monitor instead of having to add a new memory monitor with the new rendering.
    - Right-justified numbers in table view
    - In "signed integer" rendering, you should be able to enter hex numbers with "0x".
    - Need to change endianness of hex mode.
    - Expressions should be used for the memory monitors in case the actual address of the expression changes – comment from Samantha is that the model is responsible for this.
    - Model problem – "&main" should really just be "main", since functions are already addresses.
    - Scrollbar – should the scroll bar represent the entire memory region or do page-up and page-down. Maybe the range of the scrollbar should be configurable.
- Desire is to have some of these fixes in patches to 3.1, but IBM team won't do any more renderings.

TI – Open Debug Server Proposal – Paul Gingrich

- See Paul's presentation – "Open Debug Server"
- Would like to create a Debug Server with pluggable components that we can have a default implementation and companies can write their own enhanced components as IP.
- This is a replacement for GDB that is more device-software centric.
- TI would be interested in contributing some of this code.

- This would start out as a framework where we define the interfaces to the pluggable components. It's not yet clear how much implementation would be supplied.
- This would be done in Java, not C.
- Concerns
    - Need a Debug Server that doesn't require a GUI. Not sure a headless Eclipse would have the performance.
    - Performance in Java probably wouldn't be good.
    - There is some overlap with CDT now, which is probably ok.
    - In which project does this belong? DD, CDT, new project?
    - Debug Server should eventually plug into the Eclipse Debug interfaces directly.
    - Parallel tools guys want this.
- Next steps?

Editor integration
- Should core debug help integrate "default" editor with debug function?
- Breakpoint view and groups
- Do the current "Global" breakpoints satisfy the requirements of
DSDP implementers?

- Are breakpoint groups in WB 3.1 sufficient?
    - Most people had not tried them yet
    - Samantha gave a demo
    - Can show based on type, files, projects, resource working sets.
    - Can work on breakpoints as a group.
    - Need:
        - Breakpoints by launch configuration.
        - Breakpoints by debug context.
        - Import / Export
    - Breakpoints groups are called "Working Sets" for consistency. Concern is that this naming hides the feature from customers because it's not intuitive.
- Global breakpoints
    - Many folks really don't like that all breakpoints are global. Companies have taken steps to workaround this.
- Has there been any discussion of adding Working Sets to the register view (to selectively pick registers to look at.)
    - Alain: there is already the concept of custom groups.
- AI: Need use cases for breakpoints by launch and debug context. IBM to post some use cases and other companies to respond.
- Editor
    - Getting custom stuff onto a user-specified editor is a challenge. If the user picks a different editor (like SlickEdit), then the custom additions don't appear.
    - Should the base editor framework have knowledge of the debugger?

- Most companies are using the CDT editor for debugging. WR has its own editor.
- AI: IBM to write up what they've tried in the past for editor-debugging integration…extension point proposal. WR to have an editor expert also look at this.
- Felix: It's not just debugging. It's also static analysis. We need a generic extension solution for any editor.

Debuggee interaction
- Handle slower connections, handle serialized connections, async vs. sync
- Current threaded debug core is very difficult to connect to a debug engine that can only process commands serially
- Possible solution: have mode that forces serialized calls to debug target

- IBM Proposal: The model should report whether it's synchronous or asynchronous. If async, GetAsyncChildren() would be used instead of GetChildren().
- Darin
  - We shouldn't have two flavors. We should decide on one or another.
  - In 3.1, per-view jobs are serialized, even though they might come from different threads. Jobs can be cancelled unless they have been started.
  - You need a single entry point in the model in order to serialize the requests.
- Felix: Why have multiple jobs if they still have to be serialized? Alan: this was a fix because of the complexity of dealing with concurrent requests in the debug model.
- Alan two issues
  - Sync vs. Async
  - Can we do something while the core is busy
- Felix
  - Async – allows you to send requests from GUI thread knowing that the request is non-blocking. Allows you to send multiple requests without having to wait for response.
  - Jobs – have more overhead, and you cannot control the order of execution of the jobs, although Darin has currently imposed this for the debug model.
- There are some negatives with multiple jobs: the order of multiple jobs is hard to achieve without synchronization, and if you have synchronization, why do you have multiple jobs?
- What is better for the clients of the model?
- What is better for the model implementers?

Properties	**Multi-threaded Synchronous API**	**Multi-threaded Asynchronous API**	**Single-threaded Asynchronous API**
Responsive UI	Requires that requests are made on non-UI thread e.g. Jobs	Requests can be made on any thread	A single thread must make all requests. Since most requests comes from UI it might be the preferred thread
Concurrency	Requires one worker thread or job per concurrent request	Requestor can issue multiple requests without waiting for result	Requestor can issue multiple requests without waiting for result
Independency between views	NA	NA	NA
Synchronization between views e.g. don't update variable view until selection is made	NA	NA	NA
Synchronization with-in view e.g. don't reorder request with side effects from related sources	Requires API lock or capability of making composite requests	Requires API lock or capability of making composite requests	Implicit synchronization since all requests are made by a single thread

- Current problems
    - o Don't access the model from sync blocks – can cause the UI to block.
    - o With jobs making synchronous requests, the model doesn't see all requests and therefore can't optimize access based on all requests. It would be better if the model implementer could optimize the job scheduling to deal with duplicate requests. Right now the core handles the job scheduling and doesn't expose this to the model.
- Felix: the views are by nature asynchronous. Others: Not in the cases, where multiple target accesses are required to populate a view.
- Recommendation to create an async API to test some of this out. People seem to see a need for both.
- Building a synchronous interface on top of an asynchronous one is simple, but the other way around is very hard.

Jobs – have them today because API's are synchronous, and we have to avoid synchronous calls from the GUI.

Next Steps

- Where are we going to do this stuff?
    - o Recommendation is to do this in DD sub-project for now.
    - o Shuffle code via email prior. Next meeting, we will discuss CVS organization.
    - o If we branch core, how do we keep it in sync? Darin would need to code review, too, especially on API changes. Buy-in to changes is very important, too.

    - We could put experimental things into core in 3.2. Flexible hierarchy is one example. It would be good to experiment in our own sandbox first, though.

- Action item summary:
    - Wind River
        - Wind River, QNX (Mikhail), Darin, TI (Paul) – Sub-group
            - Flexible hierarchy prototyping and async/sync
            - Use Darin's dummy debug engine
            - Will need to keep Samantha up-to-date for the updater policy
        - Doug: Make sure TM meetings don't coincide with DD.
        - Doug: Chris R would like a face-to-face meeting on TM – maybe coincide with our next DD face-to-face.
        - Doug: Talk to Greg Watson and try to make sure he's included as an FYI on these discussions.
        - Doug: Serial terminal – what do we need to do to base this on the 3.1 console? Can we make the ANSI emulation, telnet support, and communication method (console, serial, network) pluggable?
        - Doug: Check into IP review and check-in process prior to next meeting.
    - Alternate view rendering and more fine-grained control of existing rendering.
        - IBM, WR (Ted)
    - TI
        - First stab at interfaces for Open Debug Server. Will get feedback via email, since we probably won't have time to discuss in detail at the next meeting.
    - IBM
        - Updater policy design and implementation
        - Editor/debugger extension point design
        - Memory view usability issues
        - Breakpoint working set use cases / launch persistence
    - Nokia (Ken Ryall), QNX (Alain)
        - Debug console requirements and initial design

Next Meeting
- Early October
- Toronto

<u>Topics for next meeting (stuff we didn't get to)</u>
Model vs. view driven (i.e. who is driving the view)
Customizing views for specific contexts
- Customize existing view content for views supplied by debug core
Possible presenter: Wind River

- Responses to view use-cases by Darin (via phone)

Scalability of Debug view (Presenter: Wind River)

- How to control the context for content of views
- How can we help a user work with many processes?
- Filters
- Visual queues (Use of colors was suggested)

Other topics

- Clone and pin
- CDT views – should be a goal to create a pluggable views in core debug. Shouldn't need to take all of CDT to get a particular view (like registers)