Minutes of discussion, October 29, 2014
Attendees: Dani, John, Tyler, Alex, Max, Martin, Wayne (remote)

Draft vision

Cloud-based development is becoming a reality, but thirty years of investment is desktop tooling means that it will remain a significant part of developers' solutions for the foreseeable future.

Our vision is to build leading desktop and cloud-based development solutions, but more importantly to offer a seamless development experience across them. Our goal is to ensure that developers will have the ability to build, deploy, and manage their assets using the device, location and platform best suited for the job at hand. To deliver this vision will require new micro-service based architecture.

In the shorter term, the Eclipse platform and related projects needs additional focus and resources on meeting the expectations of Java developers in particular. Quality, performance, out-of-the-box experience, Maven and Gradle support, and close affiliation with new Java releases are all part of the solution.

To protect against poor quality plugins ruining the developer experience, we need to instrument and monitor for performance and quality defects and link them back to the installed feature causing them, allowing the user to uninstall problematic features to improve quality. This will help improve developer experience and put pressure on low quality plugins to improve. Quality scores can also be published in the marketplace to help end users make better choices.

High level discussion topics

We read through the draft vision statement and largely agreed with the content. The element of providing monitoring/protection/awareness of cause of problems was missing but we could work that in.

We cannot finalize and publicize the vision until there is a resourcing/staffing plan to work on it. Our job is to come up with the list of technical priorities and assess the resource requirements, then bring this to the foundation staff and board to solve the resourcing side.

We need to build a list of the big ticket priority projects and then seek the resources to solve them, rather than being constrained at the start by the resources on hand.

We hit a roadblock when an adopter problem requires the technical experience from committers but the problem itself is not felt in the core platform. The group with vested interest in solving the problem does not overlap with the group in a position to solve it.

- Can we setup feature teams that combine committers with community members impacted and have them all commit to work together on solving a problem

We want Eclipse to be the best Java IDE on the market. What do we need to do to get it there

- Out of box experience, polish, quality
- Differentiator tooling features
- Ongoing support - legacy support

Desktop Platform

Can we protect against, gather data on, and/or inform the user about problematic features they have installed:

- This plugin is taking a long time, do you want to remove it?
- This feature is causing a lot of errors, do you want to remove it?
- Monitor startup time, UI responsiveness, and provide feedback to user so they can make more informed decisions about what tools to use.
- Similar to browsers which inform you about plugins causing problems and offer to remove them.

Some discussion of integration problems between Maven and JDT. Dani thinks there may be a mechanism available to address their classpath problems. Dani and Max to follow-up separately

Can we do more first class integration of third party build technologies such as Grunt, Maven, etc, directly integrated into the platform build model, convert build problems into error markers, etc. CDT has infrastructure for parsing console output and converting to markers, provides a lower barrier to external builder integration.

JavaScript tooling. Need to higher bar of JavaScript support in Eclipse on desktop. Integrate support for npm for JS build/package.

Language agnostic tooling. Generic highlighting based on simple pattern matching. Offers a low bar of support for a large set of languages. Alternatively the xtext approach of automatically generating a low level of language tool support based on a simple language model.

When Eclipse cannot find the right type of editor, the last resort is asking the operating system to open a system editor. This rarely turns out to be what user wants. We can make a product-level configuration change to open the generic Eclipse text editor by default.

Java 9 is still a black box. It could bring major changes that require a lot of work, it might not. We don’t know the actual timeline on when specifications will be available that let us get started on it. But we need to account for the fact that in 1-3 year horizon it will consume a lot of JDT focus and resources.

SWT is falling behind, we need a GUI toolkit that can keep up to date with native widget and operating system changes. Either catch up SWT across major platforms (Win/Linux/Mac), or

switch to JavaFX (which also has problems of falling behind so it's not clear it mitigates the risk). For the browser there is no single rendering engine that we can use across all major platforms. This was WebKit for awhile but there is no longer a reliable WebKit for Windows.

GTK+ 4 is coming along with Wayland, another disruption for Eclipse/SWT on Linux. Two separate ports is one possibility to mitigate risk for stable stream as new stream evolves. However this incurs overhead for the development team to maintain two ports.

Cloud Platform

Micro-service JDT. Remove single workspace/user assumption, split out indexing, refactor UI dependencies out of tooling services to enable them to be exposed as cloud services. Defining APIs and protocols for tooling web services to enable ecosystem of language tools.

Flux file sync as an enabling technology to start connecting desktop tools to cloud services. Is there a common open implementation of the message bus that arbitrary services can connect to, or will there be a selection of private message buses that only have a curated set of services available.

There is a virtuous circle that cloud tool providers will have a strong motivation to continue adding tooling value because it will drive more tool compute usage and there is a direct connection to revenue.