CHAPTER 24

# Enabling Productive Software Development by Improving Information Flow

Gail C. Murphy, University of British Columbia, Canada
Mik Kersten, Tasktop Technologies, Canada
Robert Elves, Tasktop Technologies, Canada
Nicole Bryan, Austin, Texas, USA

At its core, software development is an information-intensive knowledge generation and consumption activity. Information about markets and trends are analyzed to create requirements that describe what a desired software system needs to do. Those requirements become information for software developers to use to produce models and code that, when executed, provide the behavior desired for the system. The execution of a system creates more information that can be analyzed as to how the software performs, and so on.

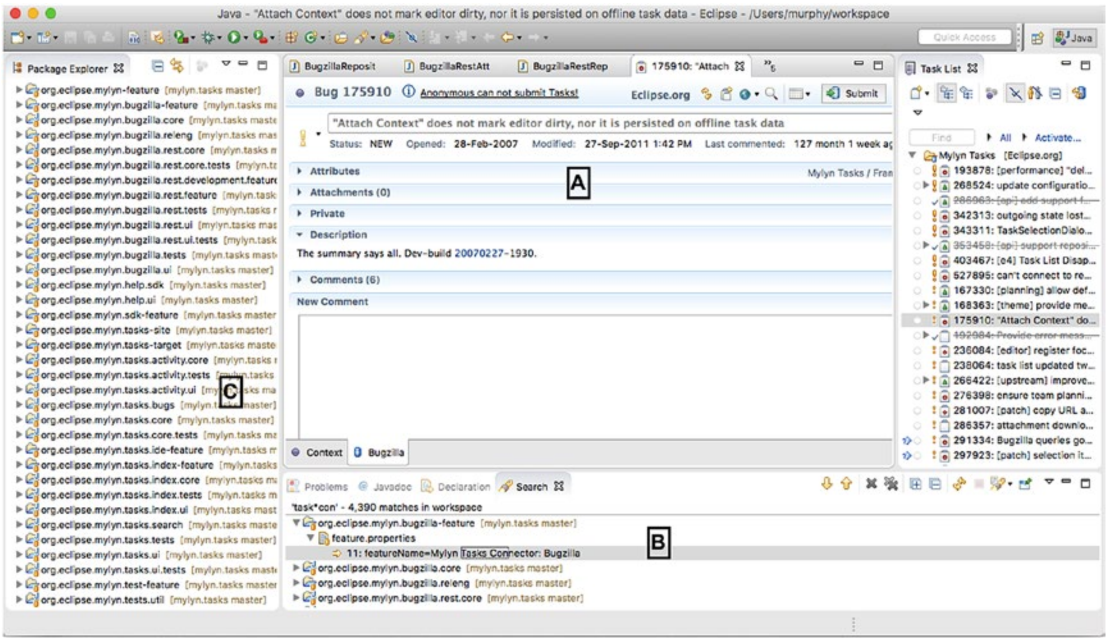
We are interested in how software tools can enable the productive development of software. Our hypothesis has been that software development productivity can be increased by improving the access and flow of information between the humans and tools involved in creating software systems. In this chapter, we review an evolution of technologies that we have introduced based on this hypothesis. These technologies are in use by large software development organizations and have been shown to improve

C. Sadowski and T. Zimmermann (eds.), *Rethinking Productivity in Software Engineering*,
https://doi.org/10.1007/978-1-4842-4221-6_24

software developer productivity. The description of these technologies highlights how productivity can be considered at the individual (the Mylyn tool), team (the Tasktop Sync tool), and organizational levels (the Tasktop Integration Hub).

## Mylyn: Improving Information Flow for the Individual Software Developer

A software system cannot exist without code that executes to provide the behavior of the software system. To produce code for a system, a software developer must deal with an amazing amount of information, such as written requirements, documentation about libraries and modules, and test suites. The result for a developer can be information overload. Figure 24-1 shows a snapshot of an integrated development environment as a software developer works on a bug fix. The developer is consulting a description of the bug (A), the other hidden tabs in the main portion of the screen hold source code already accessed as the developer is investigating the bug, the result of a search on a portion of a method name described in the stack trace is shown in the bottom part of the screen (B), and the left side provides access to the many bits of code making up the system (C). Within this environment, to produce code for a new feature or a fix for a bug, the developer must perform many navigation steps to access the contextual information needed. The friction just to get started on a task can be significant. The more complex the system, the more information a developer may need to find and cognitively maintain to start work on the task. If the developer worked on only one task a day, the friction might be manageable. However, studies have shown that developers, on average, work on approximately five to ten tasks per day, spending only a few minutes at any one time on a particular task before switching to another task [3]. As a result, developers constantly spend time finding, and re-finding, the bits of information they need to work on a task, impeding their productivity.



Java - "Attach Context" does not mark editor dirty, nor it is persisted on offline task data - Eclipse - /Users/murphy/workspace

- org.eclipse.mylyn-feature [mylyn.tasks master]
- org.eclipse.mylyn.bugzilla-feature [mylyn.tasks ma
- org.eclipse.mylyn.bugzilla.core [mylyn.tasks maste
- org.eclipse.mylyn.bugzilla.releng [mylyn.tasks mas
- org.eclipse.mylyn.bugzilla.rest.core [mylyn.tasks m
- org.eclipse.mylyn.bugzilla.rest.core.tests [mylyn.ta
- org.eclipse.mylyn.bugzilla.rest.development.featur
- org.eclipse.mylyn.bugzilla.rest.feature [mylyn.task
- org.eclipse.mylyn.bugzilla.rest.tests [mylyn.tasks r
- org.eclipse.mylyn.bugzilla.rest.ui [mylyn.tasks mas
- org.eclipse.mylyn.bugzilla.rest.ui.tests [mylyn.task
- org.eclipse.mylyn.bugzilla.tests [mylyn.tasks mast
- org.eclipse.mylyn.bugzilla.ui [mylyn.tasks master]
- org.eclipse.mylyn.help.sdk [mylyn.tasks master]
- org.eclipse.mylyn.help.ui [mylyn.tasks master]
- org.eclipse.mylyn.sdk-feature [mylyn.tasks master
- org.eclipse.mylyn.tasks-site [mylyn.tasks master]
- org.eclipse.mylyn.tasks-target [mylyn.tasks maste
- org.eclipse.mylyn.tasks.activity.core [mylyn.tasks r
- org.eclipse.mylyn.tasks.activity.tests [mylyn.tasks
- org.eclipse.mylyn.tasks.activity.ui [my sks ma
- org.eclipse.mylyn.tasks.bugs [mylyn.t master]
- org.eclipse.mylyn.tasks.core [mylyn.tasks master]
- org.eclipse.mylyn.tasks.core.tests [mylyn.tasks ma
- org.eclipse.mylyn.tasks.ide-feature [mylyn.tasks m
- org.eclipse.mylyn.tasks.index-feature [mylyn.tasks
- org.eclipse.mylyn.tasks.index.core [mylyn.tasks ma
- org.eclipse.mylyn.tasks.index.tests [mylyn.tasks m
- org.eclipse.mylyn.tasks.index.ui [mylyn.tasks mast
- org.eclipse.mylyn.tasks.search [mylyn.tasks maste
- org.eclipse.mylyn.tasks.tests [mylyn.tasks master]
- org.eclipse.mylyn.tasks.ui [mylyn.tasks master]
- org.eclipse.mylyn.tasks.ui.tests [mylyn.tasks mast
- org.eclipse.mylyn.test-feature [mylyn.tasks master
- org.eclipse.mylyn.tests.util [mylyn.tasks master]

BugzillaReposit | BugzillaRestAtt | BugzillaRestRep | 175910: "Attach

Bug 175910  Anonymous can not submit Tasks!  Eclipse.org

"Attach Context" does not mark editor dirty, nor it is persisted on offline task data

Status: NEW  Opened: 28-Feb-2007  Modified: 27-Sep-2011 1:42 PM  Last commented: 127 month 1 week ag

- Attributes  Mylyn Tasks / Fran
- Attachments (0)
- Private
- Description

The summary says all. Dev-build 20070227-1930.

- Comments (6)

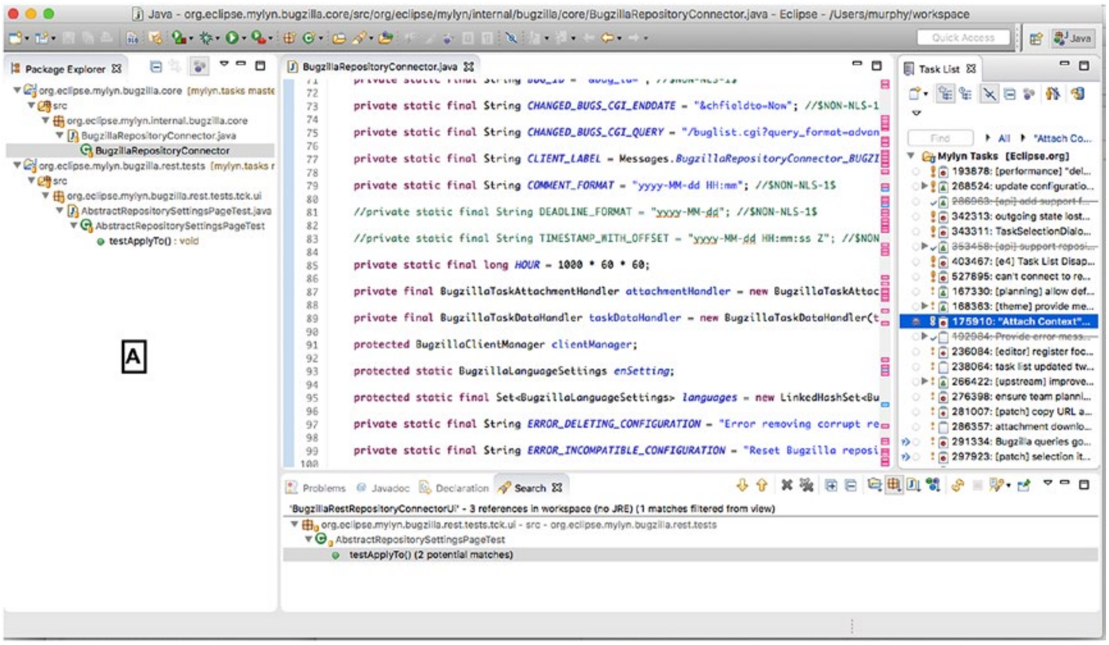
New Comment

Mylyn Tasks [Eclipse.org]
- 193878: [performance] "del...
- 268524: update configuratio...
- ~~286963: [api] add support f...~~
- 342313: outgoing state lost...
- 343311: TaskSelectionDialo...
- ~~353458: [api] support reposi...~~
- 403467: [e4] Task List Disap...
- 527895: can't connect to re...
- 167330: [planning] allow def...
- 168363: [theme] provide me...
- 175910: "Attach Context" do...
- ~~192984: Provide error mess...~~
- 236084: [editor] register foc...
- 238064: task list updated tw...
- 266422: [upstream] improve...
- 276398: ensure team planni...
- 281007: [patch] copy URL a...
- 286357: attachment downlo...
- 291334: Bugzilla queries go...
- 297923: [patch] selection it...

'task*con' - 4,390 matches in workspace
- org.eclipse.mylyn.bugzilla-feature [mylyn.tasks master]
  - feature.properties
    - 11: featureName=Mylyn Tasks Connector: Bugzilla
- org.eclipse.mylyn.bugzilla.core [mylyn.tasks master]
- org.eclipse.mylyn.bugzilla.releng [mylyn.tasks master]
- org.eclipse.mylyn.bugzilla.rest.core [mylyn.tasks master]

***Figure 24-1.*** *Information overload in integrated development environment*

To address these points of information flow friction for an individual software developer, we created the Mylyn task-focused interface for integrated development environments [2]. Mylyn changes the paradigm with which a developer interacts with the artifacts making up a software system by framing a developer's work explicitly around the tasks performed. With Mylyn, a developer begins work on a task by activating a task description. A task description may be a description of a bug or a new feature to develop in an issue tracker. Once a task is activated, Mylyn begins tracking the information a developer accesses as part of the task, modeling the developer's degree of interest in information using an algorithm based on the frequency and recency with which information is accessed. For instance, if a developer accesses a particular method definition only once as part of a task, as work on the task progresses, the interest level of that method in the degree-of- interest model will reduce. If another method is edited heavily by the developer as part that task, the interest level will remain high. These degree-of-interest values can be used in several ways. For example, the model can be used to focus the development environment on just the information that matters for a task. Figure 24-2 shows the development environment interface when focused on the same bug-fixing task introduced earlier. In this view, the development environment provides easy access to just the information that the developer needs for the task being worked on: all other information is easily accessible but does not visibly clutter the screen. As a result, the developer can see how the information accessed fits into the structure of the system (A) and has easier access to the parts when needed. Behind the scenes, as a developer works, Mylyn is automatically modeling the information flow and is surfacing the most important parts of that flow in the interface for easy access. This model can then be used to flow information into other development tools. For example, the active task can automatically populate commit messages for SCM systems such as Git. Or it can be attached to an issue to share with another developer, allowing the information accessed by one developer to another developer doing a code review for that same issue.


Java - org.eclipse.mylyn.bugzilla.core/src/org/eclipse/mylyn/internal/bugzilla/core/BugzillaRepositoryConnector.java - Eclipse - /Users/murphy/workspace
Quick Access
Java
Package Explorer
BugzillaRepositoryConnector.java
Task List
Find
All
"Attach Co...
Problems
Javadoc
Declaration
Search
A

org.eclipse.mylyn.bugzilla.core [mylyn.tasks maste
src
org.eclipse.mylyn.internal.bugzilla.core
BugzillaRepositoryConnector.java
BugzillaRepositoryConnector
org.eclipse.mylyn.bugzilla.rest.tests [mylyn.tasks r
src
org.eclipse.mylyn.bugzilla.rest.tests.tck.ui
AbstractRepositorySettingsPageTest.java
AbstractRepositorySettingsPageTest
testApplyTo() : void

```
    private static final String CHANGED_BUGS_CGI_ENDDATE = "&chfieldto=Now"; //$NON-NLS-1

    private static final String CHANGED_BUGS_CGI_QUERY = "/buglist.cgi?query_format=advan

    private static final String CLIENT_LABEL = Messages.BugzillaRepositoryConnector_BUGZI

    private static final String COMMENT_FORMAT = "yyyy-MM-dd HH:mm"; //$NON-NLS-1$

    //private static final String DEADLINE_FORMAT = "yyyy-MM-dd"; //$NON-NLS-1$

    //private static final String TIMESTAMP_WITH_OFFSET = "yyyy-MM-dd HH:mm:ss Z"; //$NON

    private static final long HOUR = 1000 * 60 * 60;

    private final BugzillaTaskAttachmentHandler attachmentHandler = new BugzillaTaskAttac

    private final BugzillaTaskDataHandler taskDataHandler = new BugzillaTaskDataHandler(t

    protected BugzillaClientManager clientManager;

    protected static BugzillaLanguageSettings enSetting;

    protected static final Set<BugzillaLanguageSettings> languages = new LinkedHashSet<Bu

    private static final String ERROR_DELETING_CONFIGURATION = "Error removing corrupt re

    private static final String ERROR_INCOMPATIBLE_CONFIGURATION = "Reset Bugzilla reposi
```

Mylyn Tasks [Eclipse.org]
193878: [performance] "del...
268524: update configuratio...
~~286963: [api] add support f...~~
342313: outgoing state lost...
343311: TaskSelectionDialo...
~~353458: [api] support reposi...~~
403467: [e4] Task List Disap...
527695: can't connect to re...
167330: [planning] allow def...
168363: [theme] provide me...
175910: "Attach Context"...
~~192984: Provide error mess...~~
236084: [editor] register foc...
238064: task list updated tw...
266422: [upstream] improve...
276398: ensure team planni...
281007: [patch] copy URL a...
286357: attachment downlo...
291334: Bugzilla queries go...
297923: [patch] selection it...

'BugzillaRestRepositoryConnectorUi' - 3 references in workspace (no JRE) (1 matches filtered from view)
org.eclipse.mylyn.bugzilla.rest.tests.tck.ui - src - org.eclipse.mylyn.bugzilla.rest.tests
AbstractRepositorySettingsPageTest
testApplyTo() (2 potential matches)

***Figure 24-2.** Mylyn's task-focused interface active in integrated development environment*

To determine whether Mylyn helps improve productivity by giving developers access to information when it is needed, we conducted a longitudinal field study. In this study, we recruited 99 participants who were practicing software developers using the Eclipse integrated software development environment. For the first two weeks of the study, participants worked with the integrated development environment as normal. The development environment was instrumented to collect logs of how the developer worked. Once the developer had reached a threshold of coding activity, the developer was invited to install the Mylyn tool within their integrated development environment. Further logs of coding activity were then collected as the developer worked using Mylyn. To ensure we could reasonably compare the activities before and after the installation of Mylyn, we defined thresholds of coding activity for acceptance into the study. Sixteen participants met our thresholds for study acceptance. For these participants, we compared their edit ratios—–the relative amount of edit and navigation events in their logs—both before and after Mylyn use. We found that the use of Mylyn improved the edit ratio of developers, adding support that Mylyn reduces friction of accessing information and improves productivity when looked at through the lens of actions performed. In other words, developers coded more, and navigated around looking for information less, when the tool focused their coding and supported their context switching. Mylyn is an open source plugin for the Eclipse integrated development environment (www.eclipse.org/mylyn) and has been use by developers around the world for more than 13 years.

## Tasktop Sync: Improving Information Flow for the Development Team

In working with organizations using the open source Mylyn tool, and a commercial version of Mylyn our company (Tasktop Technologies Inc.) produced called Tasktop Dev, we learned about additional friction for accessing information that was occurring at the team level. Increasingly, companies have been moving away from the use of one vendor's tools to support all development activities to the use of best-of-breed tools for each development activity, chosen individually by the different teams in the organization. As a result, business analysts who focus on requirements gathering may be using a tool from one vendor, the developers writing code using another vendor's tool, the testers a tool from a third vendor, and so on. While each best-of-breed tool may enable productive work, the information flow between teams is impeded as information must be manually re-entered into a tool used by another team or moved in some other form, such as via a spreadsheet or an e-mail. Information can also fail to flow, causing difficulties in the development, such as errors when a given team may not have access to needed information. With the increasing agility and need for speed of delivery in software development, a lack of automation of information flow between teams is a major impediment. A Forester survey in 2015 identified that gaps in the process of integrating tools had become the number-one source of failure and cost overruns of efforts to modernize the software lifecycle in organizations. The impact on the productivity of teams due to friction in the flow of information between teams leads to a decrease in team productivity.

Through our work on Mylyn and Tasktop Dev, we have gained expertise on the variety of ways in which tasks—a unit of work—are described in the best-of-breed tools used by different teams in large software development organizations. We realized it was possible to abstract the notion of a task across these tools and to enable automatic movement of task information between tools. In 2009, we introduced a tool called Tasktop Sync. Figure 24-3 provides an abstraction of what Tasktop Sync supports. By serving as a platform, Tasktop Sync enables the flow of task information between tools from many different kinds of teams, from the project management office through to handling service requests.

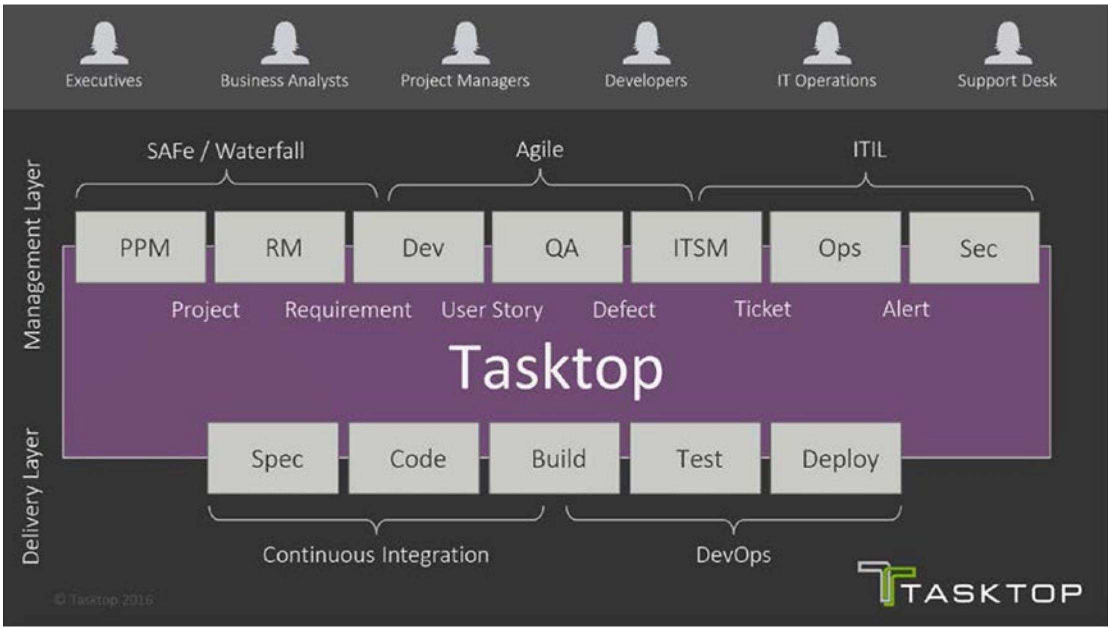


***Figure 24-3.*** *Tasktop Sync Platform view*

Tasktop Sync works in the background, synchronizing information across tools in near real time. Tasktop Sync accesses information in the tools via each tool's API. As each tool represents task information using a different schema and within a different workflow, Tasktop Sync relies on configuration information to map and transform data between the tools. For example, a task in a tool used by a business analyst may be a requirement with a short-form identifier and a longer name. When synchronized to a developer's tool, the title of the associated task in a developer's tool may become a concatenation of the identifier and the longer name from the requirements tool. The synchronization rules extend beyond simple data transformations, such as concatenation. When a data value indicates workflow status, such as whether a defect is new or has just been reopened, the status of the information must be appropriately mapped to workflow in other tools. Sometimes the matching of workflow information may require multiple changes of state of the data in another tool, such as requiring a task to move from a created state automatically into an open state.

Synchronizing information between tools also requires the interpretation and management of context of tasks between tools. In a business analyst's tool, a task (a requirement) may exist within a hierarchy. This hierarchical context must be mapped appropriately to other tools. For instance, an issue tracker used by a developer may need this information represented in an epic and user story structure. As tools can sometimes represent contextual information in multiple ways, including as links to information in other tools, maintaining context during a synchronization requires careful handling.

As software development is not a linear activity, to support teams appropriately, Tasktop Sync enables bidirectional synchronization. For instance, if tasks created by a business analyst in their tool have been synchronized to a developer's tool and the developer subsequently starts working on the task and adds a comment requiring clarification on the nature of the task, the comment can be automatically synchronized back to the business analyst's tool. Combined, these capabilities of Tasktop Sync means that a team member can work in a best-of-breed tool optimized for the work they perform, yet they can interact directly with other team members in near real time in their own best-of-breed tool choices.

Tasktop Sync has been used both within and between organizations to improve the flow of information between teams involved in a software development project. A credit card processing company used Tasktop Sync to integrate the results of tests from a testing automation tool into a tool used by the organization to chart project progress. A major automotive manufacturer used Tasktop Sync to synchronize change request and defect data between their suppliers' tools and the tools used in their organization. An important factor in the automotive manufacturer's case was the ability to configure workflow differences between multiple repositories in use in particular instances of a given tool by a supplier. The manufacturer reported times of less than three seconds to synchronize information between a supplier and themselves, providing much needed transparency between software that would be integrated into the manufacturer's product.

## Tasktop Integration Hub: Improving Information Flow for a Software Development Organization

As we have been working to improve the flow of information in software development, there have been substantial changes in the approaches taken by organizations to develop software, largely catalyzed by the DevOps movement. Over the last ten years, the DevOps movement has helped organizations consider how to increase automation in all parts of the software life cycle and to increase the focus on simultaneously achieving quality in software with faster delivery times [1]. Thinking about the overall software delivery process has led to the emergence of a consideration of the value stream of software delivery in which the delivery process is considered as an end-to-end feedback loop of flowing value to customers in a way that optimizes for business value. As a simple example, consider an organization with two software development delivery teams: one that delivers a mobile app and another that delivers a web-based app to the company's insurance business. The first team is able to deliver more customer-facing features per month than the second team. By analyzing the value stream of software delivery for each delivery team, it is determined that the mobile app team uses an automated testing process that speeds the creation of new features with high quality compared to the web-based app team. The organization may use this information to improve the software development processes across more of its teams.

At Tasktop, our products have continued to evolve. Our focus remains on improving information flow across the organization, and our latest product offering, Tasktop Integration Hub, has replaced the Sync and Dev products. Tasktop Integration Hub enables visibility across an organization's value stream of software delivery. Building on our knowledge of synchronizing data across the tools used by different teams, Tasktop Integration Hub provides insight into what information flows are occurring between different tools for different projects. Figure 24-4 shows a sample Tasktop Integration Landscape drawn automatically from the integrations various teams have set up between their tools. A landscape enables an organization to consider, and optimize, the steps that are occurring in their software development process. As it executes, Tasktop Integration Hub captures data about how information is flowing across tools used by the development teams. This data enables cross-toolchain reporting so that such aspects of development as the time to value from requirement being specified to being deployed can be tracked. The need for Tasktop Integration Hub came from the sheer number of teams and tools that an enterprise IT organization needs to connect in order to support the flow and access of information across their software delivery value streams.

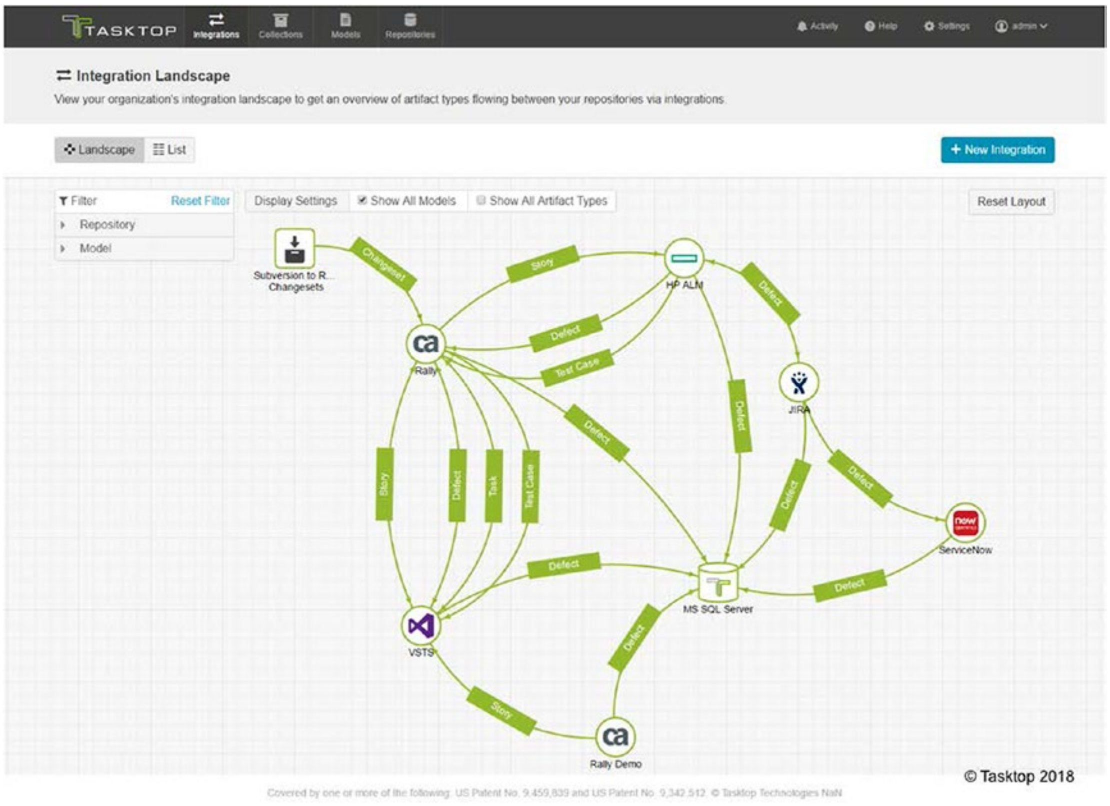


***Figure 24-4.*** *Tasktop integration landscape*

By supporting visibility into the software life cycle and by supporting an ability to track metrics as changes to the life cycle are introduced, Tasktop Integration Hub enables a determination of where friction is occurring in the life cycle, a precursor to being able to implement changes to reduce the friction and improve productivity at an organizational level.

Returning to the example of the mobile app and web-based app delivery teams within an organization, Tasktop Integration Hub provides an explicit view of how information flows across the tools used by each delivery team and can report metrics on how many customer-facing features are progressing through each of the tools used by different parts of the delivery teams. Differences between various teams in this flow of information through the value stream can be used to question different approaches being taken and to identify where there are opportunities for improving productivity through process changes, such as introducing automated testing.

## Takeaways

Delivering high-quality software quickly is the goal of many organizations, whether their end goal is a software product or whether their business relies internally on the software developed. As software is an information-intensive activity, the ability to deliver value is critically dependent on the flow of, and access to, information. When information does not flow appropriately, delivery is delayed, or worse, errors may occur, causing a decrease in quality or a further delay in delivery. If the flow of information is supported and optimized, delivery times can be shortened, and productivity within an organization can rise.

In this chapter, we have considered how information flows at different levels within a software development organization. Individuals must access particular information within the tools they use. Teams must have access to information entered and updated in the tools of other teams. Organizations must consider how the activities of different teams combine to create a value stream of software delivery. By considering these different flows and where friction occurs, tool support can be designed to help improve flow and improve productivity. We have described our journey through initial academic research, the open source Mylyn tool, and follow-on commercial application life-cycle integration products built by Tasktop, which have led to productivity improvements at the individual, team, and organization levels. Given how much software has penetrated into every kind of business, improving the productivity of creating software means improving the productivity of a vast number of businesses. Further analysis of information flow may lead to additional productivity improvements in the future that can have far reaching impacts into healthcare, commerce, and manufacturing domains to name just a few.

## Key Ideas

The following are the key ideas of this chapter:

- The flow of information among software developers is directly related to productivity.
- When the flow of information is adequately supported, delivery times on software can be shortened, and productivity within an organization can rise.
- Individuals, teams, and organizations need different kinds of support for information flow.
- Individuals, teams, and organizations can benefit from information flow that respects the best-of-breed and individual tools in which they can work most effectively.

## References

[1] Humble, J. Continuous delivery sounds great, but will it work here. CACM, 61 (4), pp. 34–39.

[2] Kersten, M. and Murphy, G.C., Using task context to improve programmer productivity. In Proc. of FSE, 2006, pp. 1–11.

[3] Meyer, A.N., Fritz, T., Murphy, G.C., Zimmermann, T. Software Developers' Perceptions of Productivity. In Proc. of FSE, 2014, pp. 19–29.