REportal: A Service-Based Web Portal for Reverse Engineering and Program Comprehension

William M. Mongan

Agenda

- Background and Motivation
- REportal 1.0 Architecture
  - Maintenance
- Re-engineered REportal
  - Maintenance
- Research Contributions
- Conclusions and Future Work
- Demo

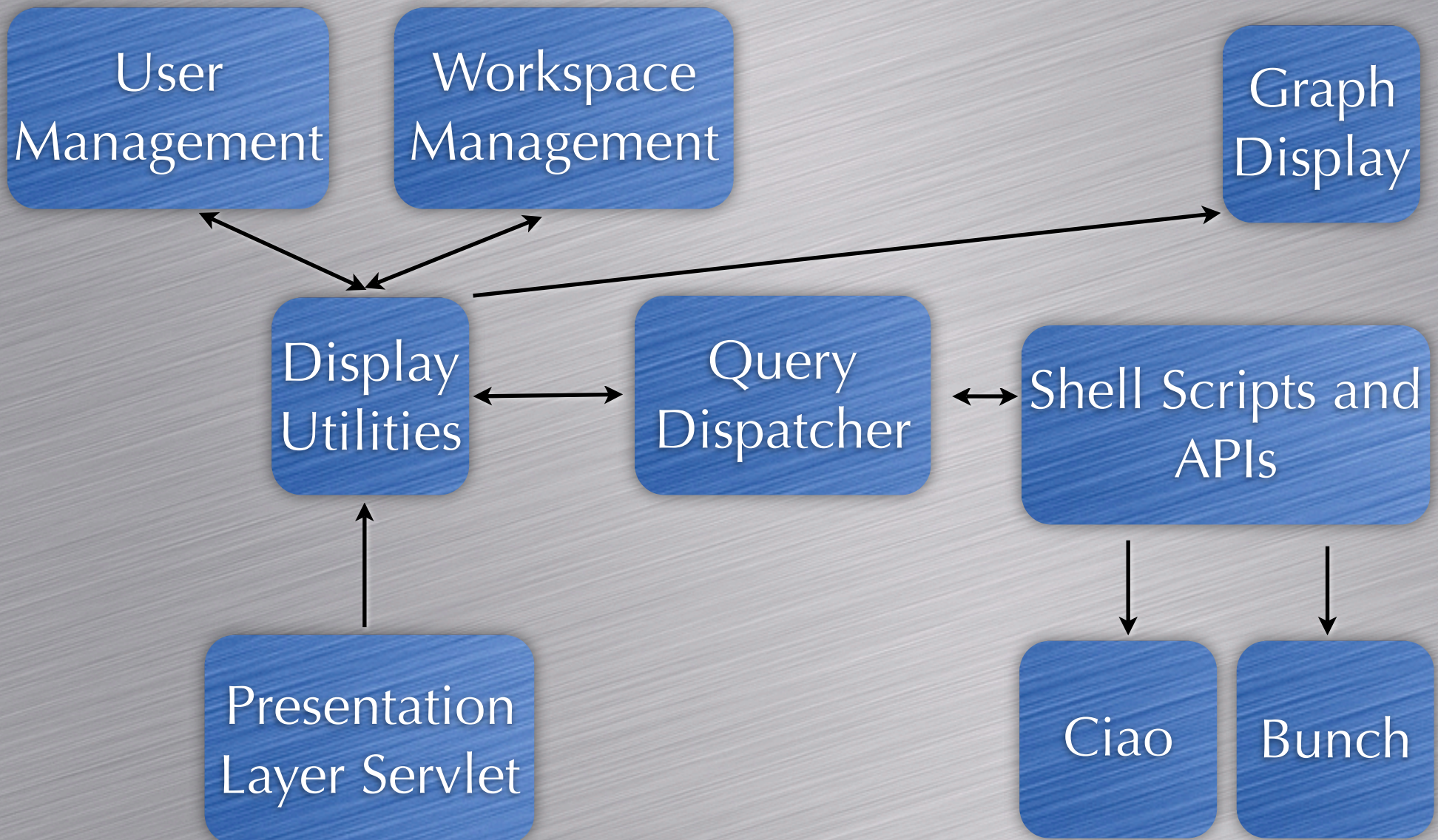
Background and Motivation

- REportal is a service-based reverse engineering portal.
- Users may upload code to REportal and perform RE analysis, without needing to install, configure and run individual tools.
- These tools were not user-friendly, and had numerous configuration dependencies.

Challenges

- REportal 1.0 was based on Java Servlets, but the presentation layer was tightly coupled to the tools.
- The tools quickly became obsolete, and others simply didn't work, hindering the functionality.
- Due to coupling, it was difficult to update the portal.

REportal 1.0 Architecture



Maintaining REportal

- For example, suppose we wanted to add a feature to REportal.
- Because the interface is coupled to the logic that executes the tool and the tool itself, changes are needed to the presentation code ("Display Utilities" which prints HTML) and to the subsystem that runs the tool.

Maintaining REportal

- The logic that executes the tool must be written in Java, because REportal was built upon Java servlets.
  - Otherwise, one must use native calls.
- Either approach destroys portability of language and platform.
- They turn "Display Utilities" into a processing unit rather than a presentation layer mechanism.

Deploying REportal

- REportal had several dependencies on the operating system, including the filesystem.
- Changes to the system (*i.e.*, upgrading Apache) caused significant changes to the filesystem that required re-architecting.
  - For example, paths to user files would change!

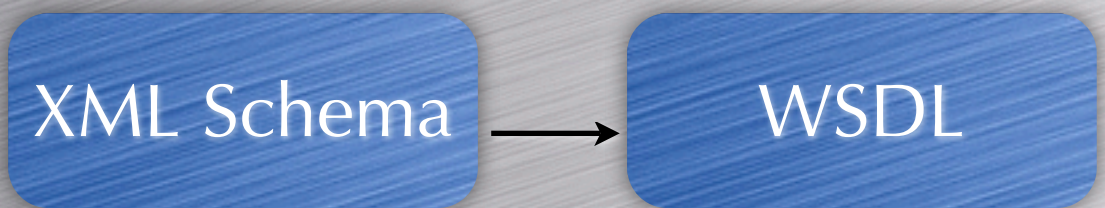
Deploying REportal

- A specific binary of Korn Shell was required to be in the web server's PATH, and a number of symlinks created by Apache had to be in-place.
- Due to the dependencies on Linux binary tools, shell scripts, interpreters, and path locations, it was even more difficult to deploy REportal on another host (or even to re-deploy it on one of our servers).
- The underlying problems: The presentation layer depended on the tools, and the tools had to be co-located with the presentation layer.

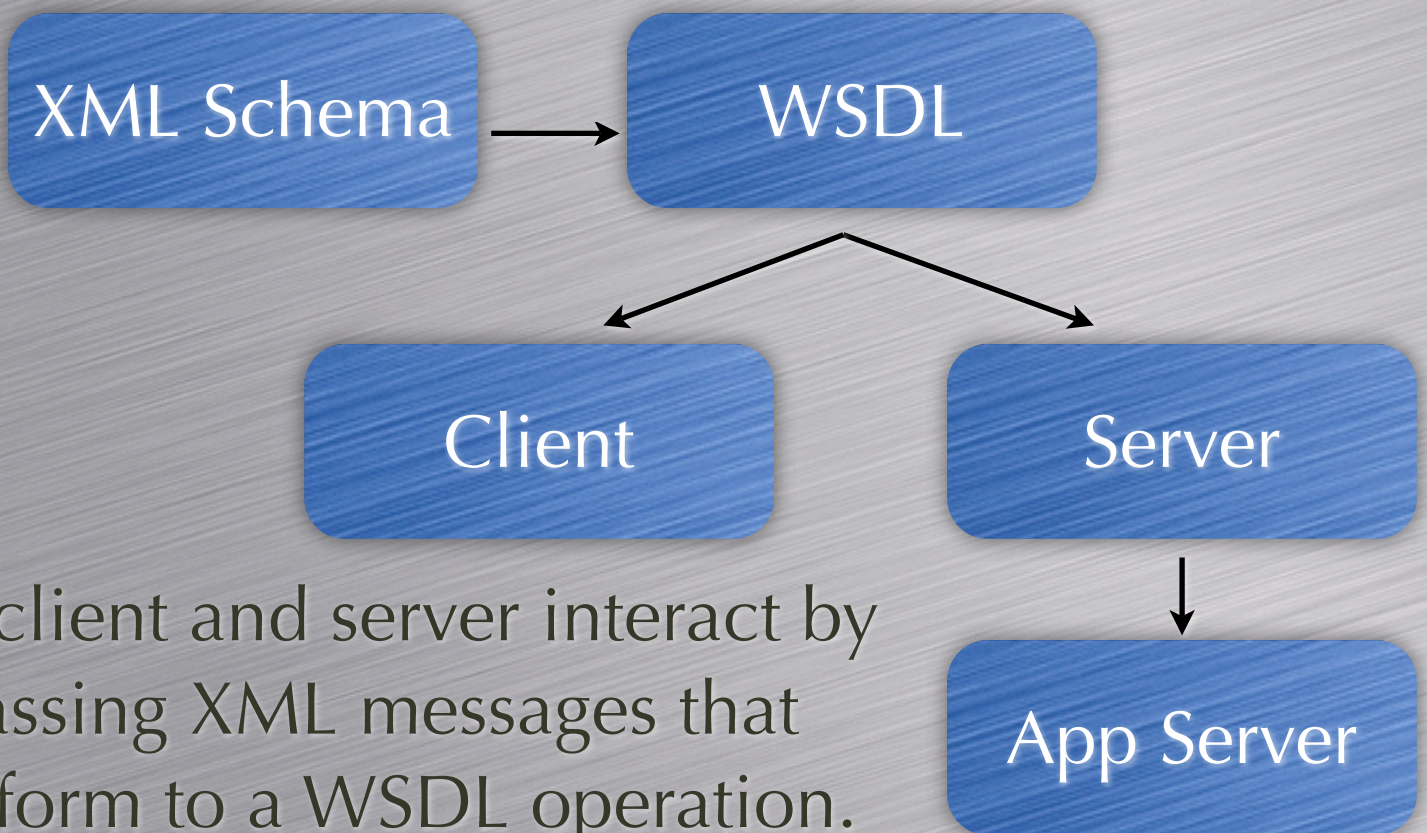
Re-engineered REportal

- REportal 2.0 is based on web services. This was chosen because the architecture decouples the interface from the tools.
- Relationships between tools are based on data via message passing in XML.
- This makes it easier to maintain the tools and to integrate legacy tools through service wrapping.

Service-Oriented Architectures

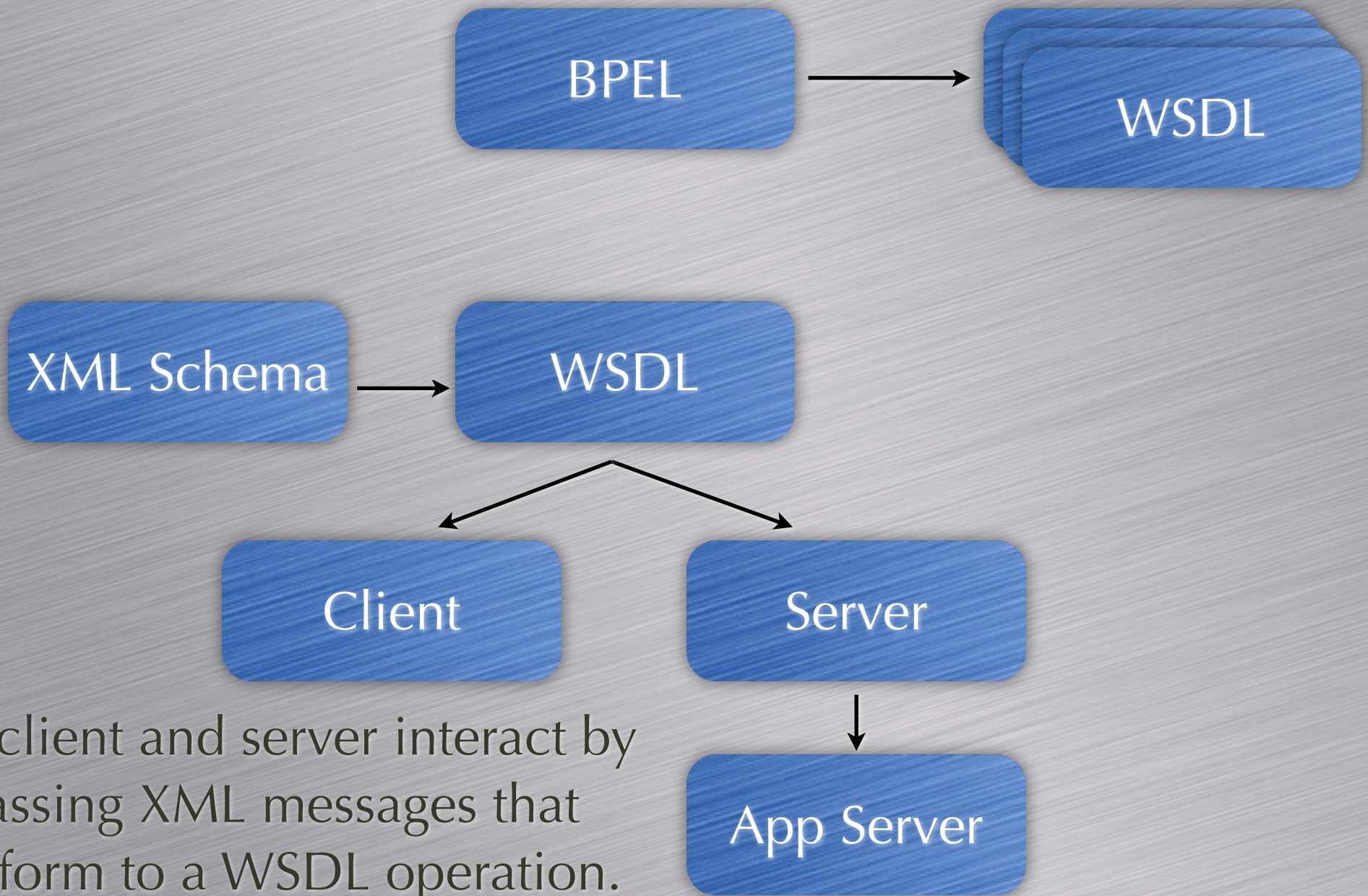


Service-Oriented Architectures



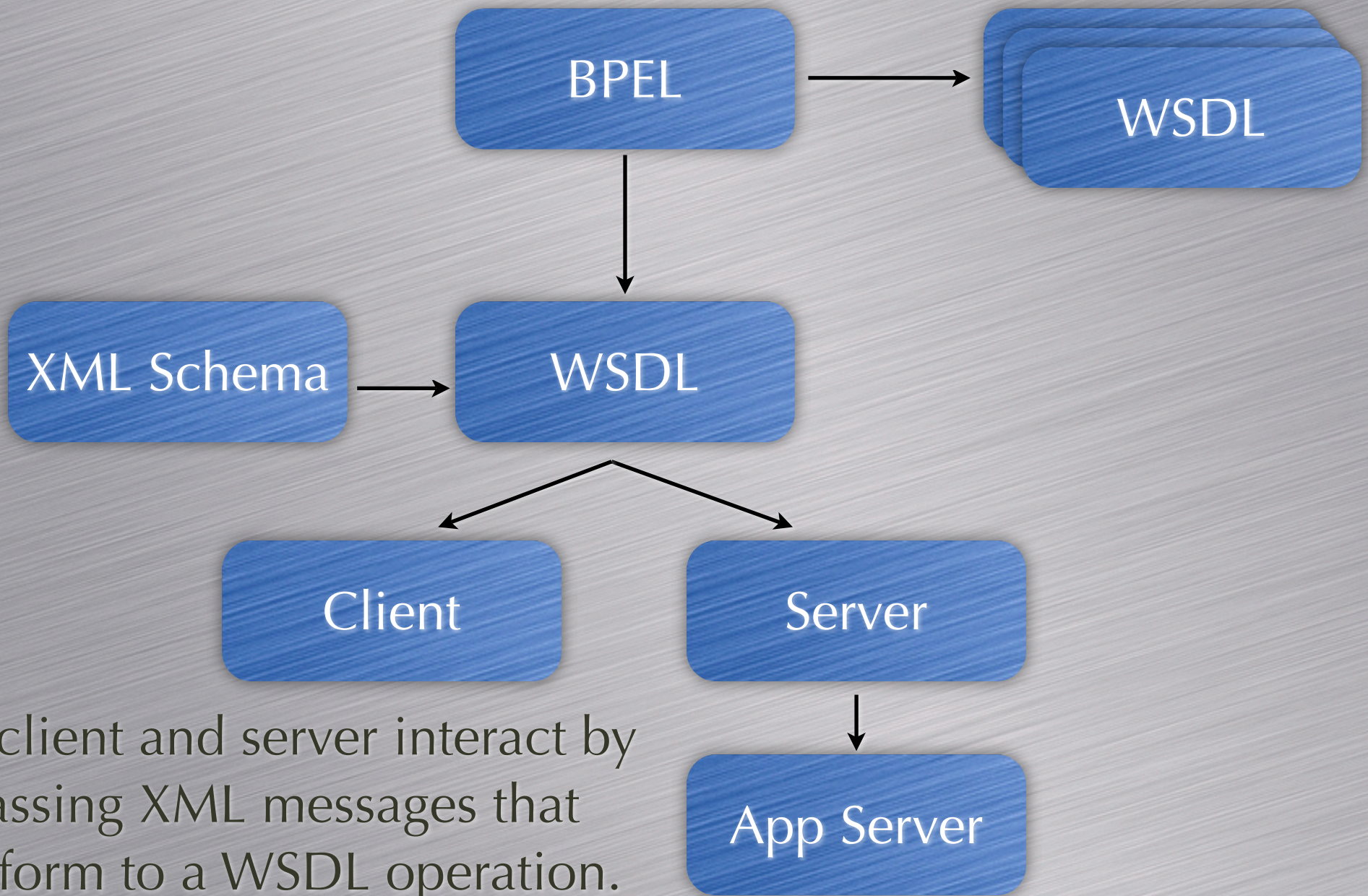
The client and server interact by passing XML messages that conform to a WSDL operation.

Service-Oriented Architectures



The client and server interact by passing XML messages that conform to a WSDL operation.

Service-Oriented Architectures



The client and server interact by passing XML messages that conform to a WSDL operation.

Benefits of SOA

- SOA decouples the services from the client. To change services, just send a new XML message to a new location, etc.
- This decouples the tools from the client - the fundamental challenge with the original REportal.
- The service implementations are language- and platform-independent; they can be deployed on any host without impacting the client.

REportal 2.0 Architecture

- The client is a thin presentation layer, implemented using JSP.
- The services expose the essence of the tools they wrap as WSDL interfaces.
- The client invokes these interfaces as it renders the graphical or tabular report in the display.

REportal 2.0 Architecture

REportal Presentation Layer includes JSP Web Pages and handles User Session State.

REportal 2.0 Architecture

Bunch Clustering Service invokes the Bunch Clustering tool on MDG's produced by querying the Static Analyzer repository.

Static Analyzer Service creates an XML repository from the code. This service also provides an interface to query the repository and obtain a JDOM object result, which is represented in an MDG.

Dynamic Analyzer Service produces an aspect to be woven into the user's code. The modified program produces a call trace of the executed features in MDG format.

Project Manager Service creates users and projects with uploaded code or bytecode.

REportal Presentation Layer includes JSP Web Pages and handles User Session State.

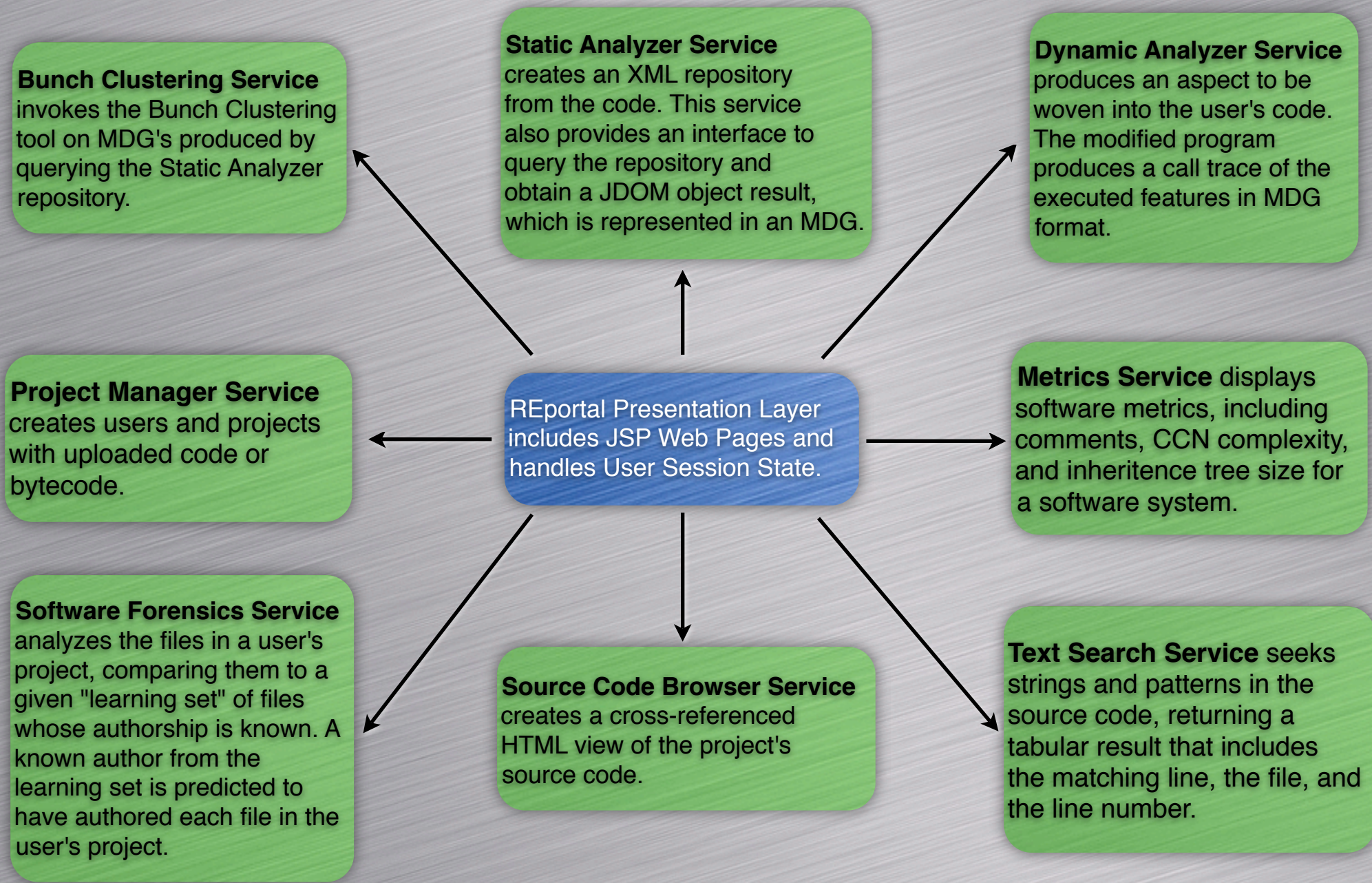
Metrics Service displays software metrics, including comments, CCN complexity, and inheritence tree size for a software system.

Software Forensics Service analyzes the files in a user's project, comparing them to a given "learning set" of files whose authorship is known. A known author from the learning set is predicted to have authored each file in the user's project.

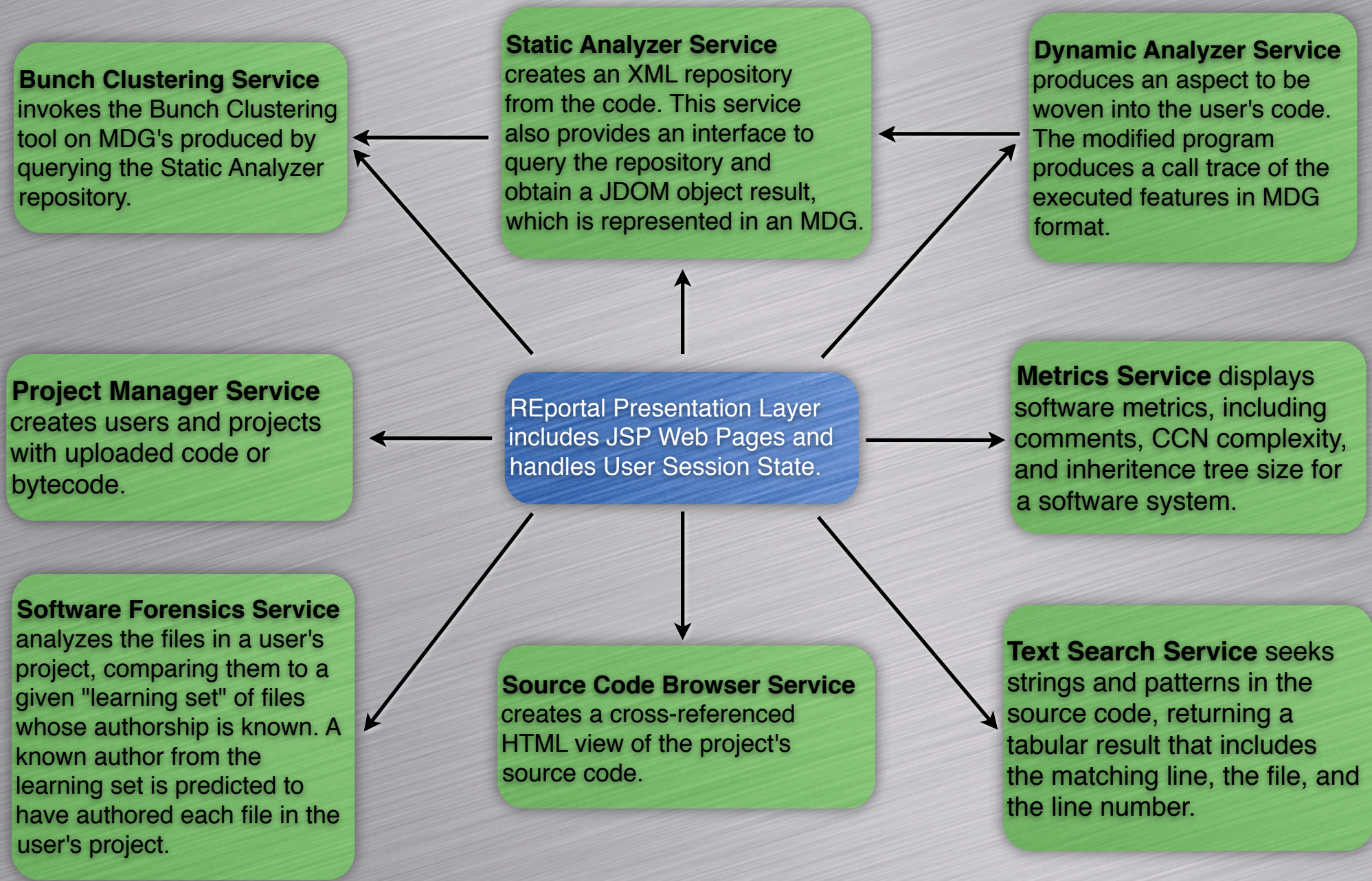
Source Code Browser Service creates a cross-referenced HTML view of the project's source code.

Text Search Service seeks strings and patterns in the source code, returning a tabular result that includes the matching line, the file, and the line number.

REportal 2.0 Architecture



REportal 2.0 Architecture



Maintaining REportal 2.0

- After developing the XML schemas, building and deploying REportal, we added a new tool as a service.
- This process was significantly easier than was previously possible.
- The tool added was a “Software Forensics” tool to determine source code authorship.

Legacy Tool Integration

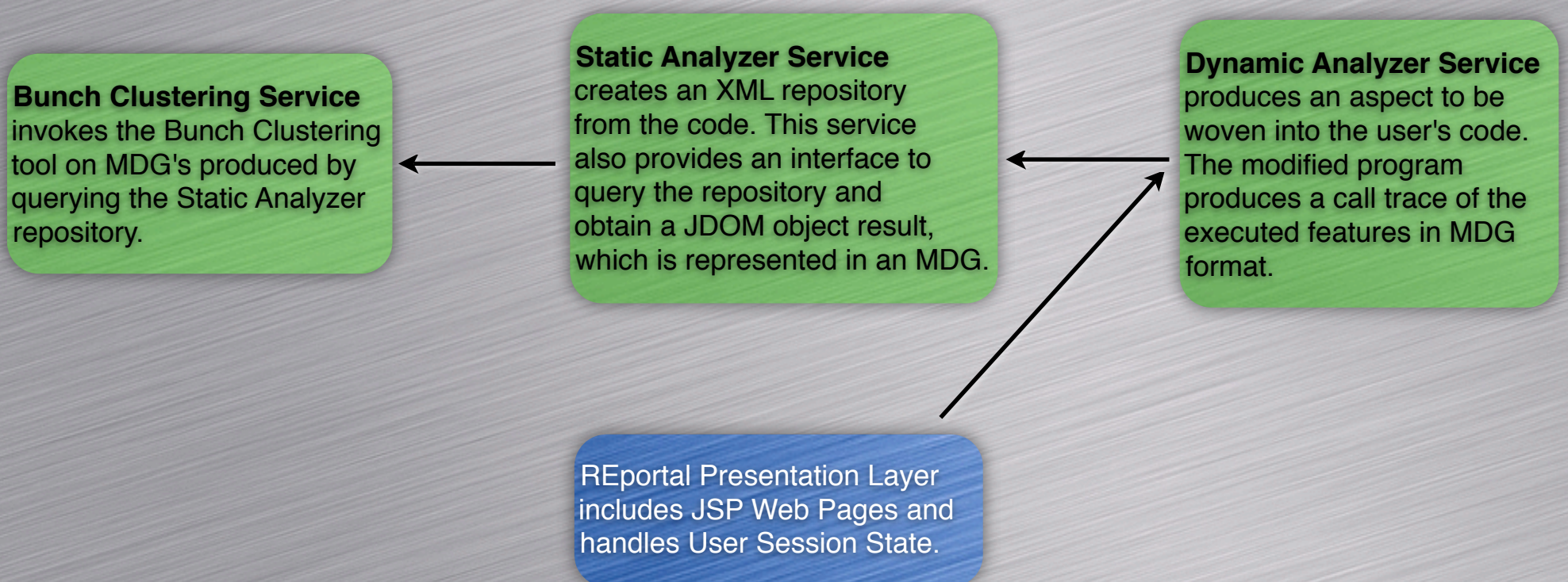
- The forensics tool works by
  - Computing metrics on code whose author is known (computing the “learning set”).
  - Determining which metrics best characterize an author.
  - Computing metrics on code whose author is not known (the “testing set”), and predicting the author based on these characterizations.

Legacy Tool Integration

- Using the existing REportal XML Schema, we use the user's project as the testing set.
- The user is only required to upload a learning set, which is a ZIP file of sample code, with one directory per author.

Research Contributions

- Because REportal integrates tools as XML-based services, we developed XML schemas for program comprehension.
- For example: "how was this feature built?"



Research Contributions

- BPEL defines service composition.
- Abstract BPEL processes do not specify actual services to invoke, but lay out a process template that can be filled in.
- In this way, services can be dynamically found, bound, and consumed at runtime.
  - This is an open problem
- As a result of this effort, one could create RE tool sequences at runtime, using data that we already have.

Conclusions

- XML-based tool integration has enabled the REportal tools to utilize one another to obtain more thorough program comprehension data.
- Deployment of the client is much easier, and development of new clients is possible.
- The SOA architecture of REportal facilitates new tool integration with fewer changes to the architecture or presentation layer.

Conclusions

- Moreover, REportal is an example instance of an architecture for general tool integration.
- Although it provides the practitioner with easy access to practical RE tools, it also represents a study in the evolution and architecture of tool-based web portals.

Future Work

- This architecture evolution can be taken further.
- We would like to automate the addition of new tools by automatically:
  - Wrapping the tool into a service,
  - Binding values from our project database (schema) to the services' inputs and outputs, and
  - Generating an abstract BPEL, mapping it to the service inputs/outputs, and invoking it.

Questions?

Approach to Tool Integration

- We inspect the front-line functions of the tool to determine its high level interface.
- Then we write a service that mimics this interface with an XML Schema that provides the required inputs and outputs.
  - Ideally, some of these inputs and outputs come from the existing REportal XML Schema.
- The service is implemented and deployed, and the client is integrated into the REportal thin front-end.

Service Integration

- Services are wrapped around the individual RE tools, which have varying platform requirements.
- Since services pass XML, they need only agree on the data types that must be shared between layers.
- This gives rise to a standard XML schema for program understanding that can be used to wrap additional tools quickly.

Service Testing

- Unit testing tests at a feature-level.
- User interface testing validates the web browser display and interface. This is important as different browsers render differently.
- Service-level testing tests at the message-passing level. XML messages are sent to the service to invoke it, and the response message is valdiated.

Service Composition

- Services are composable (like objects) to create new applications according to business processes.
- Unlike objects, services are also registered with a name service that can be searched.
- Composition is described by another web service definition: the Business Process Execution Language (BPEL).