Abstract
The caBIG™ Compatibility Review System (CRS) is a web-based application to support compatibility reviews, which certify that software applications that pass the review meet a specific set of criteria that allow them to interoperate. The CRS contains workflows that support both semantic and syntactic reviews, which are performed by the caBIG Vocabularies and Common Data Elements (VCDE) and Architecture workspaces, respectively. The CRS increases the efficiency of compatibility reviews by reducing administrative overhead and it improves uniformity by ensuring that each review is conducted according to a standard process. The CRS provides metrics that allow the review team to evaluate the level of data element reuse in an application, a first step towards quantifying the extent of harmonization between applications. Finally, functionality is being added that will provide automated validation of checklist criteria, which will further simplify the review process.

Introduction
The National Cancer Institute (NCI) is developing the cancer Biomedical Informatics Grid (caBIG™), a biomedical informatics infrastructure that will expedite the cancer research community's access to bioinformatics platforms that integrate diverse data types and support interoperable analytic tools. To help accomplish this the caBIG program has made interoperability between data and software components a primary strategic goal.

The caBIG program is composed of four domain workspaces: Integrative Cancer Research, Tissue Banks and Pathology Tools, Clinical Trials Management Systems, and In Vivo Imaging. The software applications that are developed through these workspaces must meet a set of established criteria to ensure that they are interoperable. Two cross-cutting workspaces, Vocabularies and Common Data Elements (VCDE) and Architecture (ARCH), are responsible for defining and ensuring interoperability across caBIG technology products.

Compatibility Requirements and Reviews
The compatibility review is one of the most important functions performed by the VCDE and ARCH workspaces. These reviews ensure that all applications that are certified by caBIG meet specified criteria, thereby facilitating interoperability between and among applications.

Applications developed by the domain workspaces are expected to meet certain guidelines that address issues related to the representation of, access to, and exchange of data between biomedical informatics resources. The VCDE and ARCH workspaces have established a specific set of criteria that applications must meet to be certified "caBIG-compatible". The VCDE workspace reviews criteria in the areas of UML (Unified Modeling Language) modeling, Common Data Element (CDE) creation and reuse, and vocabulary usage, whereas the ARCH workspace performs a similar review of the application programming interfaces (APIs) and test logs. In this way they ensure that applications meet a basic set of requirements for both semantic and syntactic interoperability, respectively.

The VCDE and ARCH workspaces have developed a process by which compatibility reviews take place. One of the most important lessons learned from compatibility reviews is that the current review process will not scale with demand, a fact that has become more apparent after caBIG transitioned from a pilot project and grew through its first Enterprise year. Furthermore, widespread deployment initiatives will result in an even higher demand for compatibility reviews as institutions submit internally-developed object models for review. Despite increased demand, the VCDE and ARCH workspaces must maintain a high level of quality in the review process.

The Compatibility Review System (CRS) was developed to address this issue by reducing the amount of time reviewers spend on administrative or
mechanical work and enabling them to focus on the more important task of ensuring that models meet the established criteria. In addition, the workflows built into the CRS ensure that compatibility reviews are conducted according to a consistent, defined process. Finally, the CRS ensures that reviewers complete a thorough review of the system, and it provides tracking and auditing of reviewer comments.

Objectives and System Overview

The compatibility review process is very labor-intensive and requires each reviewer to perform an extensive amount of administrative work, such as identifying and integrating content from several different sources within the project documentation for each criteria item that is reviewed. In addition, the review team lead must collate comments from each member of the review team into a cohesive document and prepare a set of reports that summarize the findings of the review team. Although it adds virtually nothing to the content of the reports this work, which is necessary to perform the review, can account for more than 50% of the time reviewers spend on the review.

The CRS was developed to increase the efficiency of compatibility reviews. The first release of the CRS, version 1.0, established the basic framework of the system and focused on reducing the amount of time reviewers spend on administrative work such as collecting information and preparing reports. A workflow was constructed to support VCDE reviewers, which allowed them to focus on the more important task of ensuring that models meet the criteria established for information models and CDEs, and that the semantic annotations that they contain are as accurate as possible. Version 2.0 of the CRS extended the functionality of the system to support Architecture reviews and provided a workflow for submitting a model for loading into the Cancer Data Standards Repository (caDSR). CRS 2.0 also included improved review project management capabilities and streamlined review interfaces. The features and workflows supported by the CRS are briefly described below.

System Design

The CRS is a web application whose architecture consists of a presentation layer and an object layer containing domain objects, model classes, and data access objects (Figure 1). The backend of the application is a local database. The web-based user interfaces are designed and developed in a Java/Struts framework, so the design of the system follows the Model View Controller (MVC) design approach wherein the view (i.e. the user interface) and the model (i.e. the actual application logic) are insulated from each other.

Interaction with the backend database is performed using either Hibernate or Java Database Connectivity (JDBC) based on performance requirements. Hibernate is used in cases where simple objects are to be inserted/updated in the database. For high throughput queries, JDBC is used.

Web interfaces are used to communicate with external data sources. Specifically, metadata including data elements and vocabulary terms and definitions, is retrieved from the caDSR for each model submitted for a compatibility review. This is accomplished using the caGrid Client API, which returns data in XML format. That API is also used to perform the queries that form the basis of the CDE reuse report (described in more detail below).

Features and Functions

The CRS contains five modules of functionality. The security module provides standard authentication methods. The administration module contains several functions for setting up and maintaining the system,
such as user provisioning (creating and managing user accounts) and workspace management (assigning workspace leadership roles to user accounts). Administrators may also create and maintain the ARCH and VCDE review checklists and manage the list of files required as part of the submission package. Administrators are also responsible for creating review projects on the system and assigning users to various roles on the review team.

The submission module allows developers to assemble a submission package based on the requirements specified by the administrator and then submit UML models for loading into the caDSR, a process that creates CDEs for each attribute in the UML model. Once the model has been loaded, curated, and validated, developers may then submit the model for a compatibility review. The submission module also contains functions used by the lead reviewers during the initiation of a compatibility review, including the validation of the submission package, answering portions of the checklist at a global level, setting a schedule for the review team, and assigning elements of the model (e.g., classes, attributes, CDEs) to each member of the review team.

At the core of the CRS is the review module, which is used by the review team to evaluate the model, its associated metadata, and the system’s APIs using the compatibility criteria specified in the VCDE and ARCH checklists. The elements in the UML model are displayed in a hierarchical tree to facilitate navigation among the elements, and data pertaining to the selected element is displayed in the detail frame of the page (Figure 2). The review page also displays the checklist for the review, which allows the reviewer to view both the data and the checklist simultaneously. Furthermore, checklist items are dynamically filtered so that only those items that pertain to the type of element that is selected (e.g., class, attribute, association) are displayed. Reviewers evaluate each element that is assigned to them and indicate whether the model has or has not met each of the relevant checklist items. Reviewers may also add general comments about each element or specific comments pertaining to any checklist item. The comments added by the reviewers are included in the reports generated by the system.

The report generation module uses predefined templates to create reports for compatibility reviews. Three types of reports currently exist, two of which

Figure 2. Review page, showing the element tree (left), data frame (upper right), and checklist frame (lower right).
summarize the findings of the review team and one that provides a way for the review team to evaluate the level of CDE reuse within the model. Specifically, the overview report and the detail report list each checklist item or element in the model, respectively, along with the corresponding comments made by the members of the review team. The generated reports may be downloaded by the lead reviewers and edited offline, then uploaded to the CRS and attached to the review project. The CDE reuse report lists each CDE in the model, and for each component of the CDE (e.g., object class, property, value domain) the report lists the number of times that component was used in other CDEs that are registered in the caDSR.

**Review Workflow**

Many of the features described above are utilized only during specific stages of a compatibility review. Therefore, they have been organized into a defined workflow that leads members of the review team through the review process. This helps to ensure that each review follows the same standardized process, it formalizes the "hand off" of responsibility at the end of each stage, and it provides a means for tracking the status of compatibility reviews as they progress through the workflow.

The review workflow contains five phases: assembling the submission package, creating the review project, initiating the review project, performing the review, and concluding the review. Each of these phases is described briefly, below.

**Assembling the Submission Package**

The review workflow begins when a developer creates a new submission package, comprised of all of the documentation required by the review team for a review. The submission package includes general information about the project, such as contact information and where the model was loaded in the caDSR, as well as files documenting the system, including the UML model, CDE information, API documentation, and reports summarizing usage of controlled vocabularies and extent of CDE reuse. When the submission package is complete the developer may submit it to the CRS administrators and request a compatibility review.

**Creating the Review Project**

When a developer requests a compatibility review, control passes to a CRS administrator. The administrator reviews the submission package to ensure that all required files are present and may return the package to the developer if errors are found. When the package is complete, the administrator forwards the review request to the sponsoring workspace lead. The workspace lead may approve or deny the review request. If the request is approved, the administrator assigns a team for the review, which includes a review project lead, lead reviewers for the VCDE and ARCH reviews, and VCDE and ARCH reviewers.

**Initiating the Review Project**

After a review project has been created, control passes to the review project lead, who creates and uploads to the CRS an XMI (XML Metadata Interchange) file containing the curated CDEs derived from the model under review. This step ensures that the information given to the review teams is up to date and provides a common starting point for the VCDE and ARCH reviews, which proceed independently but in parallel.

Once the project lead uploads the XMI file, separate review projects are created for the VCDE and ARCH review teams and control passes to the respective lead reviewers. They ensure that the content of the submission package is acceptable and may request additional information from the developers. When the package has been verified, the lead reviewers complete the global checklist, set the review schedule, and assign tasks to the review team members.

**Performing the Review**

When reviewers are assigned tasks, they may access the review project and begin reviewing the elements of the model (e.g., classes, attributes, and associations) that are assigned to them. All of the information required by the review team to perform the review is displayed on the review page or is readily accessible through links and tabs. The CRS requires that each checklist item be answered for each assigned element before reviewers may submit their final comments, ensuring that no checklist items or model elements may be inadvertently overlooked.

**Concluding the Review**

After all comments have been submitted, the project lead may generate draft reports that summarize the findings of the review team. To support reporting, two templates, which summarize comments by either checklist item or by model element, may be used. The draft reports may then be downloaded in MS Word format for editing. When the final reports are complete, the project lead may upload them to the CRS and attach them to the review project.
Evaluation and Feedback

The CRS has been evaluated by several different user groups throughout development, from a small focus group of experts to entire workspaces. At each stage feedback was collected, reviewed, and incorporated into the following release.

Specifically, early in development subject matter experts from the VCDE and ARCH workspaces, as well as from the NCI caDSR UML loading team, were consulted during the requirements gathering phase to help identify high priority features. Those people were also asked to provide feedback on the design of the user interface and workflows, since they are not only very knowledgeable about the compatibility review and UML model loading process but will also become CRS power users.

Subsequently, a pilot study was conducted in a production setting using the first public release of the system. A team was assembled to perform a compatibility review for an average size model and training sessions were given prior to the start of the review. Feedback was collected throughout the process and several usability issues were identified, which were addressed in the following release.

Finally, the most recent release of the CRS was presented for user evaluation and feedback through a demonstration and hands-on training session at the ARCH/VCDE face-to-face meeting in January 2008. Several feature requests were collected that represent enhancements of the current functionality, and they will be considered for inclusion in a future release.

Current Status

The CRS project has completed two full development cycles, resulting in an application that supports both VCDE and ARCH compatibility reviews. Additional enhancements to the system are essential, however, and a third round of development that will include expanded review functions, additional support for automated validation of checklist criteria, and administrative improvements is being planned.

Specifically, the next release of the CRS will provide additional features that will enable reviewers to assess CDE reuse and model harmonization more thoroughly, a requirement that will become increasingly important as caBIG begins to perform Gold level compatibility reviews (https://gforge.nci.nih.gov/projects/gold/). Additional automation, such as performing correspondence checks between model documentation (e.g., UML model vs. CDEs vs. published API) to verify that the system is internally consistent would reduce both tedium and human error. In addition, as applications that were previously certified as caBIG-compatible are revised it will become necessary to re-review those applications to ensure that the new release still meets the compatibility criteria. To avoid unnecessary delays in the review process it will be important to identify which elements of the model have changed since the last compatibility review, and therefore tooling will be required to perform that task. Finally, administrative enhancements will include features such as providing a high-level summary of the status of each active review in the system.

Conclusion

In summary, the CRS is important for validating that candidate applications meet criteria to achieve caBIG™ Silver-level compatibility. The CRS not only standardizes the review process but also facilitates, tracks, and automates many manually intensive processes. Furthermore, it evaluates the level of semantic interoperability through the examination of CDE reuse. This feature is the first step toward the automated identification of candidate CDEs for reuse, a major requirement for Gold-level compatibility. This may be used not only during review but also at an earlier stage, when application developers are designing and semantically annotating their model. Thus, the CRS will be a key component of caBIG Gold-level application development and will be a critical checkpoint to ensure syntactic and semantic interoperability on the caGrid2.

Acknowledgements and Availability

The authors would like to thank Brian Davis, Baljeet Dhindhwal, Amit Doshi, Lewis Frey, Prashant Kadam, Tahsin Kurc, Patrick McConnell, Salvatore Mungal, Kiran Pinnamaneni, Denise Warzel, and Claire Wolfe for their contributions.

This work was funded by contracts within the NCI caBIG Program. The CRS software is available under the terms of the caBIG Open Source Software License. Source code and documentation is at https://gforge.nci.nih.gov/projects/compatrev/.

References