



Challenges of Performing Software Configuration Management in a Service Oriented Architecture

By Amit Sheth
3/1/2008

Traditional Software Configuration Management tools only provide version control of the components in a system and do not necessarily identify the relationships between these components. Most CM tools such as Rational ClearCase allow you to define custom metadata, which can be used to save additional information such as relationships between objects with the use of custom triggers in a fully or semi-automated way. Other software CM tools such as AbInitio's Enterprise Meta Environment (EME) automatically builds the relationships between versioned components using a process known as dependency analysis. Dependency analysis requires the software components to be built with the AbInitio Software development tool also known as the *Co>Operating System* and version control the code base in the same EME repository.

When using Service Oriented Architecture for application development, dependency analysis becomes even more complex. *SOA multiplies the number of systems that you have to look at, and the number of application pieces, so much that the process isn't scalable^[1]*. The services in a SOA could be created using different development tools or middleware platforms. Additionally, their code bases could be version controlled using different version control tools and repositories, and lastly all of this might be supported by different application groups within an organization.

As a result, building a dependency hierarchy between all the components in a SOA becomes extremely difficult but nevertheless all the more important to ensure that the overall system stays functional while changes are introduced at different levels.

Here are a few things that should be taken into account:

1. **Identification of all the components that comprise the system.** This is the most important step. A detailed registry should be built for the entire architecture, and

should include hardware, software, network and all other components that span across the different services and platforms.

2. **Identifying owners of each component/service.** It is important to identify a business owner of each service as well as a change approver. In many cases the business owner also happens to be the person who approves change against the service. Ownership information should also be recorded in the above registry.
3. **Define relationships/dependency between the different components in the system and record it into a database.** This should start at a higher level and then filter down to the individual services and its components and sub-components. Although it would be ideal to have this dependency tree in a single database/registry, depending on how many layers of components and sub-components comprise the entire system as well as the number of platforms that it spans across, a combination of different databases could be used.
4. **Manage changes to the information on the components.** Once the relationships/dependencies between the different components are identified and recorded in a database, it should be kept up-to-date as new components are introduced in the system and changes are made to existing components. Automation is desirable to build this relationship/dependency hierarchy and then maintain it while changes are introduced in the system. Manual tracking is unreliable particularly as the system becomes more complex.
5. **Utilize the information to avoid conflicts/issues.** The development and support groups should make it a practice to regularly query the above database(s) before introducing new components and making changes to existing components. This is extremely important for shared components and objects.

If a change is being made to a text file format, all other downstream services that use the same text file, may need to update their code in-order to stay functional. This is quite challenging in a Service Oriented Architecture as the groups that support the different services don't necessarily communicate very well with each other particularly due to the fact that SOA encourages loose coupling. Take this excerpt from an article around SOA: *“service orientation has one common agreed-upon mission: the end customer’s participation. Users should be able to directly influence, control, and adapt the services according to their needs. At least for typical day-to-day situations, this should be possible without IT knowledge. Thus, service orientation potentially has a disruptive impact on the current structures.”*^[2]

In this case the service group that generates the text file should also identify and update all the downstream users of that text file. The identification and updates could be automated. Most recent version control tools such as ClearCase allow users to create custom triggers that could send notifications (via email, etc.) to affected service owners when an object of interest is updated in the repository.

If automated notification/updates are not possible, then a semi-automated approach of publishing change reports should be used. Reports should be generated and published in a known location such as a web URL on the intranet as soon as the Configuration Item that needs to change is identified. Downstream services should keep track of these reports.

6. **Implement Change Control Boards.** It is desirable to have a Change Control Board (CCB) which reviews these changes and approves them, particularly if these changes affect other services. Representatives from each service should attend these CCBs. In a large system the approval process could also be automated using a Change Control tool such as ClearQuest.

In summary, SOA introduces multiple challenges in performing SCM, which can be resolved using a combination of good SCM practices and tools with some custom automation.

References:

[1]

<<http://www.networkworld.com/news/2005/111405-truebaseline.html>>

[2]

<http://www.computer.org/portal/site/computer/index.jsp?pageID=computer_level1&path=computer/homepage/Nov07&file=gei.xml&xsl=article.xsl>

Configuration Management, Inc. (www.cmi.com) is a leading provider of Enterprise Change Management and Software Configuration Management® services. For more information please contact us at: 800-550-5058 or 732-450-1100.