Configuration Management and Component Libraries

Hedley Apperly

1.0 Component Management

1.1 Why do we need a Component Manager?

One of the primary ways to enable a successful component development process is by implementing a process that includes a configuration management environment or a component library.

Component management is not just about organizations with thousands of components. It is a useful engineering concept whatever the scale or complexity of the software project at hand. This is because it enables a process for parallel working and breaking large problems into smaller chunks. You may be involved in a small team using technical components and need a process and tool to simplify communication. Alternatively, you may be involved in a large multi-site, complex, component-based environment with hundreds of developers, and need real power from your process and supporting tools.

The key to a scalable solution is a scalable process and a scalable set of tools. The supply-manage-consume process is obviously scalable, especially since it has been proven in various industries over many years (Allen and Frost 1998). Tools that support a CBD process are significantly more difficult to find.

1.2 What is the Component Management process?

The basics of the supply-manage-consume process are not surprisingly the supply, management and consumption of components. Component supply can take the form of bought in components, demand driven component creation or the wrapping of legacy assets. Supply also implies a level of documentation or description. Management focuses on quality control, organization and availability. Finally, consumption, that includes finding and using components or their specifications.

1.3 What do we need from a Component Manager?

We not only need a Component Manager (tool and/or role) to manage the storage of components in a static sense but also to but also to enable the publication (supply) and consumption. The same is true for database client server systems. It is not the database that is useful, per se, but the applications that run on top of them. The services necessary from a Component Manager are defined in the following 3 paragraphs.

Table 1: Services for a Component Manager

<table>
<thead>
<tr>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Publish undocumented or informally designed components</td>
</tr>
<tr>
<td>• Use component specifications as starting point for design</td>
</tr>
<tr>
<td>• Publish components with specifications (produced)</td>
</tr>
</tbody>
</table>
1.3.1 Supply

1. Publish undocumented or informally designed components
   This task refers to the components coming into the CBD system from outside. This can include components from component suppliers, components from other teams (who perhaps have less rigor) and from legacy renewal. Legacy renewal can include mining for components in existing solutions. Non-component based solutions (or sub- solutions) can be ‘wrapped’ to provide a component-like interface. This shows that components can come into the loop without being specifically ‘produced’ as components.

   Many components that are acquired by an enterprise come undocumented and without supporting models. To make these components available to all those who intend to use them, the undocumented components need to provide enough information to enable the determination of the trustworthiness of the components and in the same manner as homegrown components. For COM, CORBA and EJB components, the interface definitions can be decoded fairly easily, but this will not complete the component specification. Information about the infrastructure and master design plan defined earlier in this book, as well as the functionality of the component and its internal algorithms are needed. If the component supplier supplies this information, then it is a straightforward case of transferring the component specification into the component library, while assuring all the necessary quality check are made. If this is not the case, then you should question the suitability of the component for use within your organization. If the prospective consumers of a component cannot trust it, then the likelihood of its use is low.

2. Use component specification as starting point for design
   This ‘design start point’ process occurs when a component supplier acts as a sub-contractor to a solution builder or consumer. The consumer’s component librarian needs to transfer the component specification to the supplier, so that the supplier will have a starting point (or contract) for design and development. This specification may be textual, but ideally, a number of UML diagrams would be provided. A first cut class diagram with <<interfaces>>, services, and operations (with parameters) is a very useful starting point for the component supplier, developing components to the criteria detailed in a specification.
Alternatively, a specification may be defined by the component supplier, in the hope that there will be demand for a generic component. In this case, it may also be useful to publish the specification for the component into the component management repository ahead of the development, to allow solution designers to work in parallel.

3. Publish components with specifications

Once the component supplier has completed the software component, the supplier will need to make it available. If the supplier has not already published a component specification, then it will need to publish the specification, implementation (optional), and executable at the same time. This form of publication allows consumers to locate components in a repository and then retrieve the physical component (executable, DLL, etc.). However, the component supplier may still choose to publish only the specification, that is, if the supplier plans to charge for use of the component. In this case the consumer would then the component specification should provide a pointer to the supplier company, so that the consumer may purchase the component.

If the specification for the new component already exists in the library and still represents the finished component, the two should be matched up to form a usable component. This can be achieved by tracking the version number of component specifications. This may not be the case if the component supplier has taken steps to make the component more generic and attractive. Where the specification no longer matches the finished component, the specification will need to be updated to reflect the changes. This may be achieved automatically or manually, depending upon the specific component management repository. This highlights the clear need to separate the four levels of component abstraction within the component library, while maintaining links between them.

Table 2: Component Abstraction Levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Component Specification</td>
</tr>
<tr>
<td>2</td>
<td>Component Implementation</td>
</tr>
<tr>
<td>3</td>
<td>Component Executable</td>
</tr>
<tr>
<td>4</td>
<td>Component Deployment</td>
</tr>
</tbody>
</table>

The links that need to be definable, to manage all the levels of component dependency are: a component specification may have more than one implementation (i.e. language), a component implementation may have more than one executable/DLL (i.e. build) and a component executable may have more than one deployment (i.e. be deployed on more than one node). This linking is particularly important for checking the impact of changes before they are made and tracking down where components have been used.

4. Re-publish components and specifications

The component supplier or consumer subsequently may need to make modifications to software components, which have already been published. To make these changes public, the component supplier will need to publish a new version of an existing component into the component library. This new version must also be linked to the previous versions so that consumers can track the components they use. This form of tracking is also important when components are split or merged as a component used in the past may now be provided as two interdependent components. Over time component may change shape and typically, complex components are split to ease understanding and facilitate use. If you are a consumer and have used a component that has been updated to provide new features, and has been split in two, you need to easily understand the new format. If it is hard to take advantage of updates to components, then the advantages are eroded and consumers may chose to stay with their current version.

5. Notify consumers of new components or problems

Publication and republication of components requires that the supplier notify consumers of software components’ availability. This task may be automated with technologies, such as e-mail, but there are choices to be made. New components will typically be advertised and e-mails could contain a copy of (or pointer to) the new component. If problems with components are identified or consumer feedback is required, the process will
be more like a ‘car recall’ notice. Advertising and warning are two sides of the same coin.

In summary, the component supplier only needs mechanisms for putting versions of components and their specifications into a sharable component library, getting specifications for new components from the library and notifying consumers when components change or appear. The key here is that the component library (or repository) is a neutral holding area between component suppliers and consumers. It acts as the conduit and enables the supply/manage/consume process.

1.3.2 Manage

Within the management of components the task will not flow in as natural an order as those in the supply and consumption processes. The sequence of features discussed is based upon the general setup and management tasks for a component librarian.

1. Manage library repositories

As described earlier, components will need to reside in different locations for different suppliers and consumers. Project teams building and using components will need their own local repositories. Departments may store components, as well as enterprises. Outside the enterprise, suppliers will advertise, sell and distribute components worldwide. The task of managing these repositories includes setting them up and maintaining their availability.

2. Manage library users

Once you have library repositories, you will need to control who has access to them. Certain users should not be restricted from access to certain areas: others may be able to browse, but not consume. At the upper end of the scale, users may be able to consume from, publish into, or manage component libraries.

3. Manage catalogues

Within a library, the contents need to be organized in such a way that housekeeping and searching are optimized. Component librarians will creating hierarchies of catalogues for locating components to maintain order.

4. Quality assure components

Part of the role of a component librarian should be to ensure that only quality components are readily available for consumption. The component librarian may choose to control which components are published into a library, perhaps by implementing a holding area for quality assurance, before components are moved into the area that can be accessed by consumers.

5. Manage components (specification, implementation, executable and deployment)

Management of software components is not a task; rather component management is a component library feature required, to support the four levels of component abstraction. Therefore this applies purely to the organization and protection of the published assets, which may be logical (specifications) or physical (built/compiled) components. The component library meta-model must be capable of holding distinct information about each of the four levels. It is also important that the links between each of the levels can be visualized and defined. This visualization should be provided graphically with an explorer tree and ideally, a dependency tree.
6. Make components available
Not only should components be available to users of the specific component library application or repository, but it may also be very useful to publish component information using alternative technologies such as documents, HTML and the Web. Even more useful is the ability to use these forms of published libraries in searches for gap fulfillment, using the same engine as the component repository, at the same time.

7. Versions
The final level of complexity for component library is that each of the four levels of component abstraction will need version identification. New versions of components can be created and linked to older versions. It will also be necessary to archive unused and old versions of components. This concept is similar to that of source code configuration management but refers to component specifications and completed components. These component levels are scoped well before coding starts and once testing is complete, i.e. outside the normal boundaries. This is the primary reason that source code control systems are not typically suited to component management.

1.3.3 Consume
1. Search for necessary components
To enable a solution builder to reuse, before they buy or build, a comprehensive search utility is necessary. Typically, the last thing you will know about a component you are looking for is its name. You are much more likely to know what infrastructure you will be implementing in, what business function you want to perform, what programming language you use and perhaps what services and parameters you need. These are the criteria that must be definable for searches.
2. **Gap Fulfillment**

Gap fulfillment should permit you to find components that are not quite the same as your specified search. Thus, you have the option to modify your overall solution design to accommodate a component that fulfills enough to your component specifications.

3. **Specify component specifications for tender**

If a component generally meets the consumer’s needs, the solution builder may choose to publish a component specification for tender. The form of the specification should be equal to the search criteria specified above. The published component specification will give a component supplier information about what to build and why it should be constructed. The specified information alone may be insufficient, requiring the supplier and consumer to enter into dialogue.

4. **Use or reuse component specifications**

During the design stage, once you have found a suitable component, you may choose to use the specification in your design. This is true even if the component has not been completed yet. However, if the component executable is available, when the specification is used, it should be stockpiled for the solution assembly later.

5. **Use or reuse components**

If you are less structured in your design process, or perhaps purely focused on your application build phase, likely you will need to use component implementations or component executables. This reflects the majority of current, widget level component reuse. This in no way demeans the suitability of component management for this type of reuse. The problems encountered here are typically large numbers of small components that can only be used if found. At this level, component management eases the process of communication between component vendors and direct users.
6. **Deploy Components**
When the executable components are actually deployed onto the runtime computer, deployment details should be recorded against nodes.

7. **Register interest**
Whenever components are purchased, used or libraries searched, the consumer should be able to register interest in the library or the component. This information should be used to inform the consumer when new components or important information becomes available.

8. **Receive New Component Notification**
Expecting to receive notification is the result of registering interest for new information. As identified earlier, a good way to receive notification is via e-mail, possibly with the new component embedded. At a minimum, a hot-link to the component in its repository should be provided, so that the component can be reviewed for suitability and use/reuse.

9. **Review New Components**
The new or modified component specification can now be interrogated to evaluate its worth. If no new benefits are obtainable, the message can be ignored. Otherwise, you may decide to take advantage of this component.

10. **Refresh component specifications and components**
As you have already used this component’s specification, executable, or both, then you need to integrate the new version with a minimum of disruption.

### 2.0 Conclusion

This completes the overview of the basic features necessary for a component manager, or component repository, to enable a supply/manage/consume process for CBD. As you can see, the majority of the features focus on the process of publishing, finding and using components, not just the static management of such artifacts. The component management repository acts as the builders yard, catalogue, holder of specifications, search facility, tracker for component use and the informer of changes. This is no mean feat and is something that is not purely optional in today’s CBD oriented world.

I have described the major steps that relate to the way in which suppliers and consumers want to use configuration management systems and component libraries. Therefore, standard features required from configuration management and component repository applications were described and a set of evaluation criteria was identified. These criteria are useful in the selection of configuration management and component library tools that enable CBD, rather than hinder it.

It can also be seen that these techniques apply equally to small, one or two, person projects and large inter-enterprise CBD endeavors. Thus, an important fact concerning CBD is that the tools and process you use for CBD must be scalable, in terms of application and resource complexity, if you intend to achieve success through CBD.

### 3.0 References

