Summary Review
A Testbed for Configuration Management Policy Programming
van der Hoek, André

Peer reviewed by:
Nathan Whitehead
Xiaoyu Zhang

Summary:
Van der Hoek, et al. present an architecture which separates configuration management (CM) repositories from CM policies. Based on this architecture they introduce an abstraction layer that defines a generic, distributed repository and a programmatic interface for defining CM policies. The purpose of their system, NUCM, is to provide a testbed platform on which CM policies can rapidly be generated and tested. They present three systems which have been implemented (or replicated) using NUCM.

Ratings:
Importance mean: 4.2
Importance mode: 4
Writing mean: 4.6
Writing mode: 5

Commentary:
Containment Model:
Containment modeling is a technique for illustrating the structures of a content management system's data model. For a complete definition see [1].
Drafting a containment model for NUCM demonstrated a weakness in the capability of containment modeling. The bold containment relationships shown above are relationships which the current version of the containment modeling scheme has no method of describing. The logical repository is a very important conceptual element of NUCM, but it has no explicit representation in the system. As such, the bold relationships are neither inclusive nor referential, and they are stored neither on container nor on containee. Logical repositories exist by virtue of the elements they contain and the relationships among those elements. In order to properly understand NUCM, it is necessary to understand the logical repositories. Perhaps containment modeling should be extended to include a virtual container type and an implicit relationship type.

The containment model for NUCM suggests several questions about the structure of the NUCM repository. On page 83, the authors state that "logical repositories can themselves be part of other logical repositories." Since logical repositories are not explicitly represented in NUCM, it is not clear what this means. In as much as logical repositories are defined by connected graphs of collections and atoms, it is clear that every logical repository contains other logical repositories (because every connected subgraph is a logical repository). It is not clear that this is a meaningful or useful distinction.

The containment model raises at least three further questions: can atoms exist without belonging to any collection? Can empty collections exist? Can collections contain themselves, either directly or indirectly? The first two questions are related. In order for the programmatic interface to work, it seems that either atoms must be able to exist without belonging to collections or empty collections must be permissible. Otherwise, how would collections and atoms be stored in the repository, given that the storage functions can only store one element at a time. The final question does not have a clear answer. Collections can contain other collections to allow for a hierarchical structure. The containment relationships are referential, so it is, theoretically, possible for collections to contain themselves either by direct reference, or via one or more intermediary collections. It is not actually stated, however, whether NUCM actually will permit such a cycle to exist.

**Concerns:**
Nearly all reviewers expressed a generally positive opinion of NUCM. Most liked the idea of a CM testbed and felt that NUCM provided a good one. Most of the concerns expressed were related to specific details of the version model, and probably stemmed from the minimal discussion of versioning included in the paper. The version-related reservations would likely be assuaged by a more complete discussion of why versioning support is so minimal, and of how common versioning techniques would be implemented using NUCM.

The version-related concerns included confusion about the apparent lack of a version model, confusion about how branching and merging work, and confusion about how differences are computed and changes are stored. The version model is at the heart of
any CM policy. Any definition of a version model inherently limits the CM policy; therefore, to be truly policy-independent, a CM platform cannot contain any explicit versioning support beyond a mechanism for declaring general relationships among items in the system, and a naming scheme flexible enough to accommodate version identifiers as part of item names. The reasoning behind the lack of a version model could have been explained more clearly in the paper. It also would have helped if the authors had demonstrated more explicitly how several common version models (perhaps the traditional version-tree and change-set based versioning) could be implemented in NUCM. A deeper explanation would likely have quelled the concerns related to the version-model, and to branching and merging.

The concerns about computation and storage of differences are more significant. As one reviewer points out, the list of interface functions on page 87 contains no functions which can be used to compute differences between items stored in the repository. In the strictest sense, CM can be defined to include only functionality related to storing items, versioning those items, and selecting valid sets of item versions. Under this definition the lack of a mechanism for computing differences cannot be considered a failing of NUCM. That being said, computation and display of differences between versions is of such value to users, and such a basic feature of modern CM systems that it is hard to see how any system which lacks such facilities could be useful. Furthermore, some CM policies rely on the ability to compute differences. Any CM policy based on change-sets or storage of deltas relies on the ability to distinguish differences between versions of a managed item. Without any discussion of difference computation, it is difficult to tell whether such systems could actually be implemented using NUCM. Even assuming NUCM does have undescribed facilities for computing differences, another reviewer points out that the facilities may be insufficient to support the authors' claim that NUCM can manage any type of artifact. As this reviewer indicates, other material we have read demonstrates that not all delta algorithms are effective for all file types. Specifically, delta algorithms which are effective for text may not be useful at all for binary files.

The version model and delta computation concerns were all expressed by several reviewers. In addition to these common concerns, one reviewer remained unconvinced that NUCM was truly capable of supporting all CM policies. He further questioned NUCM's utility as a platform for commercial CM systems. Van der Hoek, et al's claim that NUCM can support any CM policy is probably justified, but better examples, especially of change-set based policies would have helped substantiate the claim. NUCM's creators do state clearly that they offer NUCM as a research platform and specify that it not robust enough for commercial use. Further, they indicate that most CM policies can be implemented more efficiently with an optimized repository than they can on the NUCM platform. After proving a policy on the NUCM platform, they suggest developing an optimized version for real-life use.

Class discussion of the NUCM system also highlighted weaknesses in the distribution model. In particular, system security is weak. Each repository must be trusted to correctly manage its own objects. The paper's authors do not discuss how user authentication across the multiple servers occurs. Depending on the security protocols
present in the underlying operating system, it could be the case that each NUCM user would need an account on every machine which stores data he or she must access. Such a system would be quite unwieldy for the user.

**Suggestions:**
Several reviewers had suggestions for future work. In general, the suggestions focused on improving usability, improving robustness, or supporting additional functionality.

To improve usability, one reviewer proposed adding a front-end that would provide the user some guidance in defining policies. Although she made no specific suggestions about what form this would take, it might be possible to come up with a questionnaire which would collect information about the user's environment and make a policy recommendation based on the user's responses. Another reviewer proposes providing policy patterns from which a user could select version models, metadata (attribute) models, and possible even workflow or access control models.

The authors made no claims as to the robustness of their system. A reviewer proposes an approach that should provide improved fail-over capabilities. She suggests that each server backs up half the contents of two other servers. If any server goes down, its data would remain retrievable from its backups. This suggestion would improve robustness without requiring full redundancy of all systems.

Finally, one reviewer suggests that with minimal effort NUCM could be expanded to support groupware applications. As he points out, NUCM already has a communication system (to allow distributed storage of content) so it should require limited effort to extend the communication to allow real-time workspace updates and other groupware features.

**Reviews included in summary:**
Kai Wang  
Sonja Ellefson  
Kai Pan  
Joerg Meyer  
Jennifer Bevan  
Dorrit Gordon  
Mike Baker  
Rita Garcia  
Guozheng Ge

**Bibliography:**