OSA VM CM Synergy project Options - version 1

Requirements:

1. The OSA VMs will be independently buildable and versionable. (separate projects)

2. To be compatible the VMs must be built from the same version of the shared code. This includes the API and any shared source code. This does not specify the method used to share this code.

3. Some VMs such as NAS may not use the same change management system. As such the method of sharing source and API may vary from one VM to the next. To state this explicitly, a VM that works outside of CM Synergy will not be able to use CM Synergy subprojects as a sharing mechanism and will need to import/export shared code through another mechanism.

CM Synergy / Library Build Tools Limitations

1. You cannot store a file name or a directory name with dashes in CM Synergy. We overcome this by mapping '%%' to '-' during repository generation.

2. Changes to a subproject do not automatically pop up in other projects that use the subprojects. For library builds there is a mechanism that can update a subproject to the next version rather than branch it. This mechanism is only on for trunk builds. If the version number of the project has a dot (.) in it, then the mechanism is disabled. This will directly affect how shared subprojects are updated and used when a set of VM projects twigs. The mechanism is either on or off. When it is on, the changes checked in for one VM that uses a subproject will automatically be used in the newly created baselines of other VM projects that use the shared subproject. *Note: this is currently broken. We are waiting for a fix.*

3. When changing objects in a shared subproject used by one project, the changes will not automatically propagate to a different version of the shared subproject used by another project. As an example: Domain0-wookie and RAIDCore-wrath both use the shared fwarch subproject. If a file is checked out under fwarch-wookie, the change in version number will not been seen in fwarch-wrath until the RAIDCore-wrath or fwarch-wrath subproject is reconfigured to using a task that is associated with the newly checked-out object. Changes made to objects that are checked out will be seen in both projects.

4. It is possible to share a common subproject between the 2 projects. Example: Domain0-ewok and RAIDCore-boxcar both use instances of fwarch with the same baseline. The user can use the GUI to select fwarch-boxcar in the Domain0-ewok project. At that time any new checkouts or changes in fwarch-boxcar will be visible in both projects. Care should be taken when reconfiguring either the RAIDCore or Domain0 project as it may have unintended and unexpected effects on the shared subproject.

5. The cort tool as presently written does not cleanly support checking out /managing Domain0 projects and RAIDCore projects with the same version. Example RAIDCore-homerun is already checked out. The checkout of Domain0-homerun is noisy and not guaranteed to be correct.

6. We are not considering the use of Open reconfig to facilitate the sharing of changes across OSA VM projects under CM Synergy. It is difficult to control and produces unintended results.

Options

Option 1 – add new assets to RAIDCore project

Pro: keeps everything in sync across assets. No compatibility matrix needed.

Con: does not allow for independent versioning of VM products.

Option 2 – new projects with no subprojects

Pro: VM products can be independently versioned. No complicated management of subproject content

Con: Maintenance problem. Any sharing of source code is done at a file level. Very easy to have missing CRs. Files can branch. It can be difficult to characterize the interface version for the compatibility matrix.

Option 3 – new projects with shared subprojects

Pro: VM products can be independently versioned. Sharing is at a subproject level. If two projects share the same versions of the shared subprojects, they are in sync.

Con: Assets will appear in projects where they are not used. Delivery of content changes to projects can be complicated (where do I check in my changes?) It can be difficult to characterize the interface version for the compatibility matrix.

Option 3a – single shared subproject per team.

This is essentially option 3

Option 3b – combination of shared and unshared subprojects per team

(current implementation)

Pro: subprojects that are only used in 1 VM are clearly identified as such.

Con: Unless we add a new sharing scope for each possible combination of asset use, there will be projects where some assets in a shared subproject are unused. If we do add new sharing scope definitions, number of sharing scopes will get unmanageable and developers will need yet another decoder ring.

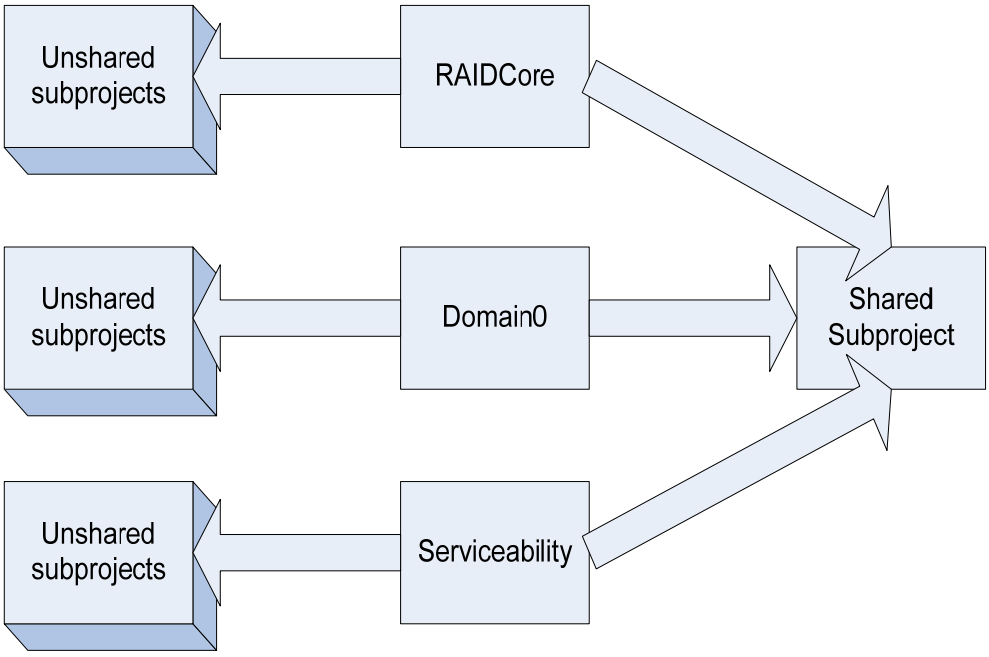
Option 3c – combination of unshared subprojects per team and shared common subproject.

(proposed new standard)

Pro: subprojects that are only used in 1 VM are clearly identified as such. Any asset that may be shared in any of the VMs will be in the shared VM. A change to any of the assets changes to version. This allows us to easily know when VMs are in sync at the interface level. We can leverage existing library build tool features to allow the shared subproject to automatically advance to the most recent version.

Con: There will be both Linux and VxWorks implementations for some assets. These assets will be present in the project work areas. The shared content for an asset team will be separate from the unshared content. Auto advance means that we cannot block changes in a VM build. Damage to one is potential damage to all. Auto advance may confuse some developers. CRs that contain changes outside the shared subproject will not be automatically picked up by other projects. If the CR is submitted to the wrong VM, the changes outside the shared subproject will be lost. The CR will only be recorded in build sheets for the first project where it is encountered.

This figure illustrates the proposed new scheme.



This change moves the content of the earlier named _linux and _osa sub-projects into the shared OSACore subproject. Additionally, assets under ios_domain0 and cas_domain0 that are proven to be shared will be moved under OSACore.

```
Domain0
  Makefile
  OSACore ← new subproject
  foundations2_domain0
  hypervisor_domain0
  platforms_domain0
```

Table 1 - New Domain0 Structure

```
RAIDCore
  OSACore ← new subproject
  foundations
  hypervisor
  platforms
etc...
```

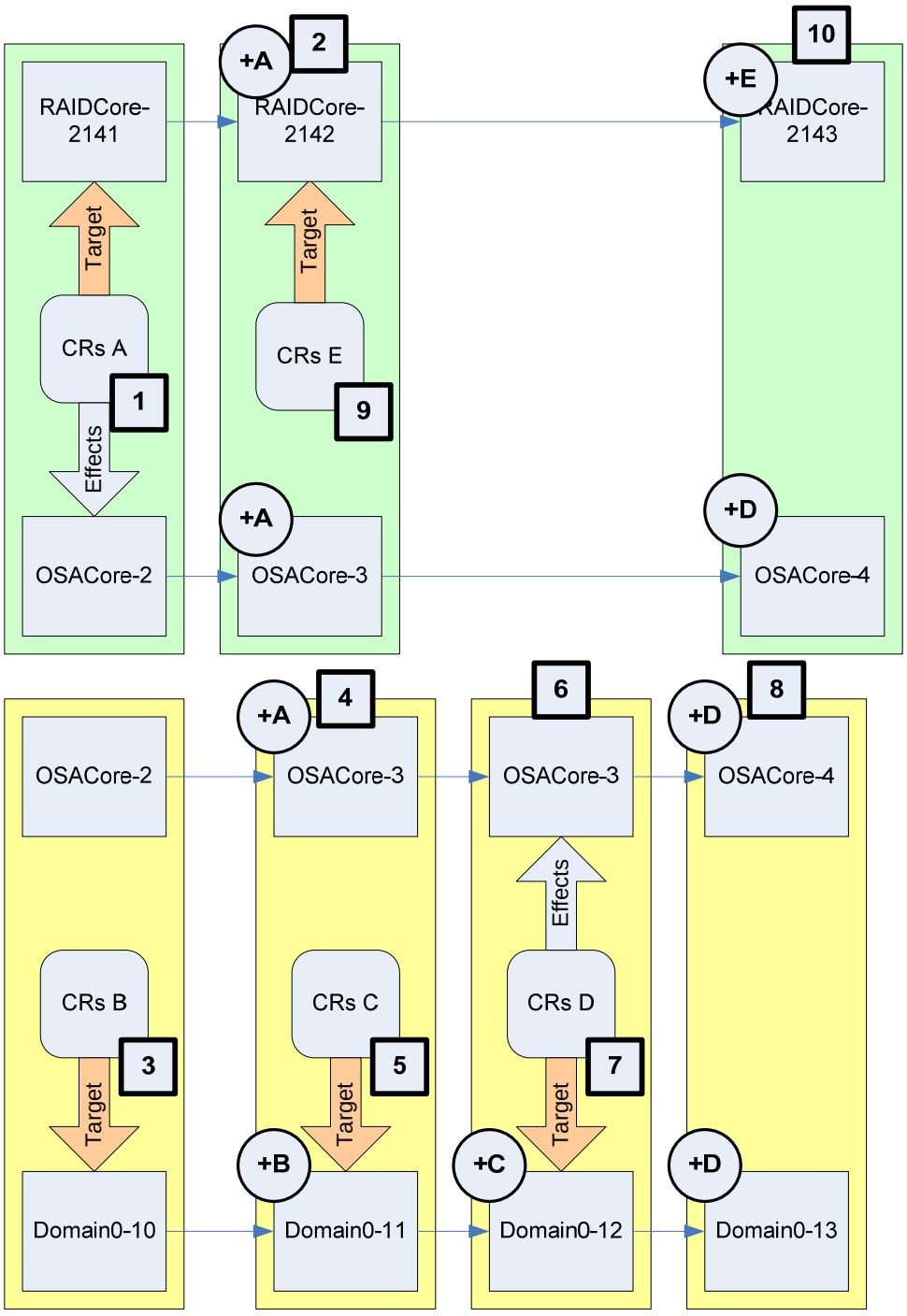
Table 2 – Modified RAIDCore Structure

The content of OSACore is organized by team names at the top level. The below that are assets and elements. Some assets will be varied for their actuation based on the OS type of the VM. Those assets will have a second level of elements named VxWorks, Linux and Common. The Common elements will be empty if there are no shared files.

```
  cas/
    rmi/
        api/
            Common/
            Linux/
            VxWorks/
        src/
            Common/
            Linux/
            VxWorks/
  foundations2/
    osa/
        api/
  fwarch/
       (shared assets)
  ios/
      ipras
        api/
            Common/
            Linux/
            VxWorks/
        src/
            Common/
            Linux/
            VxWorks/
```

Table 2 - OSACore structure

This figure below illustrates the affects of CRs on the shared subproject where automatic subproject advancement has been allowed. Specifically it shows how and when the shared subprojects will advance.



In this scenario RAIDCore and Domain0 share the subproject OSACore.

1. CR group A is targeted to RAIDCore. It affects OSACore-2
2. RAIDCore-2142 is created with changes to OSACore that produce version 3. [+A indicates that both RAIDCore-2142 and OSACore-3 have accounted for CRs A.]
3. CR group B is target to Domain0. It does not affect OSACore.
4. Domain0-11 is created. It uses OSACore-3 as the baseline for OSACore. [+B indicates that Domain0-11 now accounts for CRs B.]
5. CR group C is targeted to Domain0. It does not affect OSACore.
6. Domain0-12 is created. There are no changes to OSACore. [+C indicates that Domain0-12 now accounts for CRs C.]
7. CR group D is targeted to Domain0. It affects OSACore.
8. Domain0-13 is created with changes to OSACore that produce version 4. [+D indicates that both Domain0-13 and OSACore-4 have accounted for CRs D.]
9. CR group E is targeted to RAIDCore. It does not affect OSACore.
10. RAIDCore-2143 is created and uses OSACore version 4. [+E indicates that RAIDCore-2143 now accounts for CRs E.]