Revised Build & Release proposals

Challenges

- We continue to increment versions and release portions of the code base despite zero changes (although not as indiscriminately as in the past).
- When commencing preparation for the next general CLE release (e.g., branch trunk to 2.9.x at a given revision point) we do so from a largely unknown state with respect to changes (this is not the case with indies). Who knew what was new in 2.8.0 (https://confluence.sakaiproject.org/display/DOC/Sakai+CLE+2.8+new+features)?
- Our build and deploy process is not consistent across projects (source compile in some cares; binary deployment in others) resulting in developer confusion and frustration.
- Release early and often as a goal continues to allude us: we appear as a community capable of assembling a single CLE minor release (e.g. 2.8.0) and 3-4 CLE maintenance releases per year (and we've slipped in maintenance release output for 2011).

Year	*.0 releases	Maintenance releases	Notes
2011	2.8.0	2.7.2 (June/July), 2.8.1 (Aug/Sept?) + indie maintenance releases (e.g. kernel, etc.)	no CLE maintenance releases since Aug 2010
2010	2.7.0	2.7.1, 2.6.3, 2.6.2, 2.5.6 + indie maintenance releases (e.g., kernel, msgcntr, profile2, samigo)	
2009	2.6.0	2.6.1, 2.5.5, 2.5.4	
2008	2.5.0	2.5.3, 2.5.2, 2.5.1	The 2.5.1 release was never publicly released.
2007	2.4.0	2.4.1, 2.3.2, 2.3.1, 2.2.3	
2006	2.3.0, 2.2.0	2.2.2, 2.2.1, 2.1.2, 2.1.1	
2005	2.1.0, 1.5.0	1.5.1	
2004	1.0		

Suggested changes

Revive full source checkout and consistent build/deploy process (trunk, *.x branches)

- update svn .externals to include all core capabilities (e.g., kernel, indies, etc.) in a standard source check out.
  example: https://source.sakaiproject.org/svn/sakai/branches/sakai-trunk-all/.externals
- Provide consistent default build and deployment process rather than building from source in some instances and downloading /deploying assembly zips during deployment in other instances.
  example: https://source.sakaiproject.org/svn/sakai/branches/sakai-trunk-all/
- consolidate/eliminate or otherwise restrict to a bin-deploy profile (-Pbin-deploy) poms such as kernel-deploy and core-deploy

Leverage Maven release capabilities for all core projects (increase off-cycle release options)

- Provide release capabilities for all core projects (e.g. maven-release-plugin) in order to simplify the release process and permit greater range of "off-cycle" releases.
- Maven coordinates changes limited to <groupid> revision only (e.g., org.sakaiproject.[project]); <artifactId>, <version> will not be changed unless an overwhelmingly compelling reason exists to do so. Target trunk and 2.8.x.
- This can be done quicker than one might otherwise think.

Release project APIs independently from impl, tool modules, etc.

- Provide finer-grained, more flexible build directives
- Requires scaling back our over-reliance on parent/child relationships between poms (dependency <> parent/child). Think parent/child only in cases where modules actually share dependencies and a common configuration.
- API versioning ideal = major.minor (e.g. 2.9-SNAPSHOT, tag = 2.9).
- Start with the kernel and entitybroker.

Dependency tree refinements: add entitybroker and common projects to kernel

- Simplify the dependency tree by adding projects such as archive, import and privacy as well as the entitybroker appear to the kernel. Target trunk and 2.8.x. Consider back porting to 2.7.x.

Deploy kernel-util to shared rather than bundling it up in each project's webapp.

- Changes to kernel-util particularly to classes such as FormattedText.java can trigger tool releases since kernel-utils is bundled up in each project's webapp.

Update Maven compiler plugin to compile code using Java 6.

Focus release efforts on releasing individual projects "early and often"; consider annual releases as convenience packaging for new adopters and others.