xSDK Design Document

September 30, 2017

Stakeholders

Stakeholders for the Extreme-scale Scientific Software Development Kit (xSDK) are the ECP community (applications teams, software and technology projects, DOE computing facilities) and the broader computational science and engineering community.

Design Concerns

As explained in the document on xSDK Project Requirements, the xSDK focuses on two strategic objectives:

- **Build Community:** Reusable scientific software packages are developed in many groups. Much of this software can be leveraged by a broader collection of users if the package developers coordinate and collaborate across package teams. The xSDK see community development as a critical means of expanding the usability of individual packages.
- **Build Sustainability:** Reusable scientific software must be sustainable in order for users to rely upon it. Sustainability must be an integral part of the entire software process.

The xSDK design addresses the six types of epics of the xSDK Project Requirements:

1. Facilitate xSDK growth
2. xSDK distribution
3. Integration and use
4. High-quality products
5. User support
6. Regular upgrades

For each epic type, we explain:

- Analysis: insights about requirements
- Design: approach to addressing requirements

xSDK Epics

Epic 1: Facilitate xSDK Growth

Brief description: Processes and activities related to adding new xSDK packages.

We need a process to identify functionality gaps. Gaps can be satisfied by existing packages to integrate into the xSDK. A gap may require development of a new package. The types of stories that

would fall under this epic may include interviewing a stakeholder to determine scientific library needs, or clearly defining xSDK membership requirements for packages.

Analysis: Facilitating xSDK growth requires addressing two basic concerns:

1. **Membership:** Determining and describing a clear process for becoming and remaining an xSDK package.
2. **Scalable cost:** Keeping xSDK overhead costs minimal, such that xSDK-related activities are primarily done by each individual package teams, and these activities have high value to xSDK overall and to the package.

Design:

1. **Membership:** xSDK will have a set of evolving community policies[1] that describe what is required to be compatible with the xSDK. In addition, we will have a process for determining when a package can become an xSDK member, and how its membership is sustained, using compatibility with policies as the primary vehicle.
2. **Scalable cost:** xSDK-related activities will be primarily conducted at the package level. Very little code will exist outside of the xSDK packages themselves. In particular:
    a. Any interoperability features (one package calling capabilities in another package) will reside in the calling package. Interoperability will be peer-to-peer, not through any kind of universal interface. This includes testing capabilities.
    b. xSDK build processes will be done through a hierarchical tool (presently using Spack) such that build rules are written and supported by each package team.
    c. All release processes will rely primarily on package team support, with only a modest amount of general xSDK activities.

Epic 2: xSDK Distribution

Brief description: Policies, activities, and tool exploration/development related to distribution, deployment, and installation of the xSDK.

We have two basic types of users. Those who install for themselves and those who install for a group, including an installation at a leadership computing facility. The types of stories that would fall under this epic may include adding a candidate xSDK member package to Spack (the tool used for installing the xSDK), or setting up an installation test on a new LCF machine.

Analysis: Distribution of the xSDK as a metapackage requires a clear description of:

1. **Package-level activities** that must be completed by first-time xSDK release packages, as well as package-level activities that must be completed by all xSDK member packages for subsequent releases.
2. **Coordinated xSDK metapackage-level activities** that must be completed for each xSDK release.

Design: The xSDK will have:

1. **xSDK package release checklist**[2] to describe package-level activities that are associated with xSDK member package-level release requirements.
2. **xSDK metapackage release checklist** to describe overall coordinated release activities.

[1] xSDK community policies are available via https://xsdk.info/policies.
[2] xSDK package release checklist

Epic 3: Integration and Use

Brief description: Improve the usability and interoperability of xSDK member packages.

Best practices improve the usability of embedded scientific software and enable uniformity across independently developed packages. The types of stories that would fall under this epic may include a candidate xSDK member package achieving compatibility with required xSDK Community Policies, improving the interoperability of two or more specific xSDK packages, and adding a page to the xSDK website focused on a topic of importance to developers or users.

Analysis: We need to communicate about package status regarding compatibility with xSDK community policies, including approaches used to fulfill compatibility and progress over time.

Design: Packages working toward xSDK compatibility will track gaps and progress.[3] Each xSDK compatible package will complete an xSDK Policy Compatibility Document to indicate status and approaches to fulfilling xSDK community policies.[4]

Epic 4: High-Quality Product

Brief description: xSDK software should work properly, run efficiently, and solve the required problems.

xSDK must be bug-free on mainstream platforms and must have a comprehensive test suite to isolate errors. The types of stories that would fall under this epic may include setting up an automated test with updated compiler and/or third party library versions, adding a new xSDK Community Policy aimed at compatibility with a recognized software engineering best practice, or documenting an improved process for dealing with regressions found in packages during nightly automated testing.

Analysis: We need a clear process for testing the complete xSDK metapackage on target machines.

Design: We will specify xSDK testing plans for target machines at ALCF, NERSC, OLCF, and other commonly used platforms, including Linux and Mac OS X. We will collaborate with the broader ECP community to work toward a process for regular xSDK testing across key ECP platforms.

Epic 5: User Support

Brief description: The xSDK and its member packages should be sufficiently documented and provide clear processes for obtaining support for questions and problems.

User support includes regular updates with bug fixes, tutorials, email contacts and more. The types of stories that would fall under this epic include documenting the process by which xSDK member package teams can add information about their package to the xSDK website, setting up customer email lists, and establishing a service policy containing support expectations and commitments.

Analysis: A requirement for xSDK member packages is to provide sufficient documentation and clear processes for contacting the development team. We also need this information at the xSDK metapackage level.

[3] https://github.com/xsdk-project/xsdk-issues
[4] https://github.com/xsdk-project/xsdk-policy-compatibility

Design: We will specify xSDK plans for documentation and user engagement, with a strategy of keeping the cost of xSDK overhead minimal (see Epic 1).

Epic 6: Regular Upgrades

Brief description: Improve xSDK capabilities and make the xSDK available on new and existing computing platforms.

The xSDK must have regular releases with the latest features from each package. The xSDK should be available immediately on new platforms so that users can rely on its availability when they port to a new system. The types of stories that would fall under this epic include ensuring the xSDK is usable on a new test bed machine before application teams need access to new platforms, documenting the xSDK release process, and executing the xSDK release process.

Analysis: We need regular releases of xSDK for ECP, which are well tested on target platforms.

Design: We will determine a sustainable approach for regular xSDK releases, which incorporates the xSDK package release checklist (see Epic 2) and testing plans (see Epic 4).