Brown Dog Tool Store Design Document (BD-53)

Table of Contents

1 Overview

As part of the Brown Dog project, we need to set up a Brown Dog Tool Store to support:

- list available tools,
- search for tools,
- upload and download tools

where tools are:

- Polyglot conversion scripts,
- Medici extractors,
- Versus descriptors,
- Versus measures,
- CyberIntegrator workflows,
- etc.

This is intended as a web service. It will be first available for the ISDA group's internal use and testing, and later exposed to larger user group or even public when it is ready. The corresponding JIRA entry is BD-53.

2 Requirements

The tool store is a web service. On the web site, a user can:

- First priority:
    - list and search the tools by fields such as name, tags, author, # of downloads,
    - upload a tool,
    - download a tool,
- Lower priority:

- add tags to a tool,
- add reviews to a tool.

Additionally, an admin can:

- change the "active status" of a tool,
- delete a tool (actually changing its "active status" to "deleted").

In the Medici instance on a user's machine, "tool store" related link pages need to exist to manage the tools:

- list the tools installed on this machine/app;
- enable, disable a tool;
- install, uninstall a tool;
- choose one tool, and display its detailed info.

3 Design

We plan to do this within the Medici framework, as:

1) it's relatively familiar,
2) might be able to reuse the user accounts for identification.

Downside:

1) For the tool store, the concepts and UI items "Collections", "Datasets", "Files" and "Admin" might not apply or lead to confusion.

   We can use a special configuration where these items are not displayed on the page.

3.1 Performance Considerations

Current expectations are: the number of tools are not excessively large, and tools are not constantly added nor updated at high rates. So the design mainly focuses on features, and performance, though also important, is not at the highest priority.

3.2 Data Models

We plan to create a parent class (or Scala trait) "Tool", with the children classes of:

PolyglotConversionScript, MediciExtractor, VersusDescriptor, VersusMeasure, CIWorkflow.

Each of the individual child class could have more specific fields, for example, PolyglotConversionScript could have:

inputFormats (".stp"), outputFormats (".3ds,.obj"), usingTool ("AutoHotkey", "Sikuli", ...)

while MediciExtractor could have:

outputFormat (such as "video/mp4"), generatesPreview (a boolean).

3.3 Tool Properties

Following Apple AppStore's model, there are two levels of tool properties: tool level and version level. The details are in the link:

https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Appendices/Properties.html

Some current design thoughts are listed below.

Required:

- ID
- Name (short name)
- Description (longer)
- filename (name of the actual tool file, such as "*.py")
- Author (ISDA, a user account/ID) -- could be a list, a tool could be written by multiple persons. -- the idea of a list came from nanoHUB (https://nanohub.org/resources/BMCsuite)
- Active Status (active, archived/inactive)
- Fee Status (free, paid)
- Support Status (supported by ISDA, not supported by ISDA)
- License Type
- License Details
- Compatibility (e.g., compatible with Polyglot 3.2, or Medici 1.0) -- from Apple App Store
- Length ---- auto from MangoDB GridFS
- md5 ---- auto from MangoDB GridFS
- Current Version ---- from Apple App Store
- Date of Last Update ---- from Apple App Store, galaxy

Optional:

- Version History
- Number of downloads ---- for use in search by popularity (nanoHUB users, Atlassian Marketplace)
- Reviews
- Documentation (installation/uninstallation instructions, what's new in this version)
- Screenshots
- Wishes (wish list of features) -- from nanoHUB tool
- Ranking -- from nanoHUB
- DOI -- from nanoHUB (https://nanohub.org/resources/BMCsuite/versions)
- Questions -- from nanoHUB

- Citations -- from nanoHUB

- Language -- Apple app store, human language, but can be used as programming language for a tool (Python, Java, AutoHotkey, ...)

This is a long list. Many fields can be optional, using Scala Option's to minimize the storage size.

A design question is how can we find "Similar add-ons", where Atlassian Marketplace provides this? One solution is to search for tools with the same or similar tag values.

3.4 Author

3.4.1 Background

Currently in Medici: an email address, such as "ruiliu@illinois.edu".

nanoHUB uses "members". Each member has a profile:

First name, last name, Email, Organization, Employment Type (University/College Faculty), Web Site, Biography

Example: https://nanohub.org/members/9108

Other features in nanoHUB related to users are "Contributions", "Usage", "Favorites", and "Blog".

3.4.2 Requirements

- Need a special "author" value for ISDA, for Tool's "Support Status" field.
- Need an "Active Status" field (active, disabled, deleted, ...).

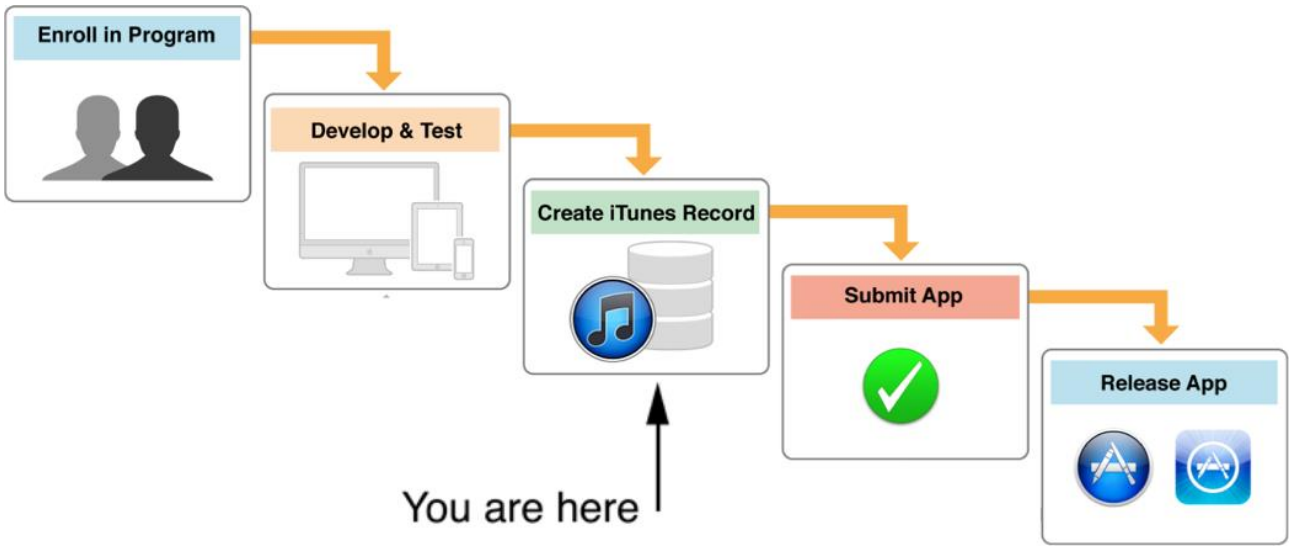
3.4.3 Design

One possibility is to extend the "socialuser.Identity" trait. For details please see http://securesocial.ws/guide/user-service.html .

3.5 Workflow

The uploading and managing of the tools require a design of a workflow. In the workflow, the status of a tool is changed, such as: created, wait to be uploaded, uploaded, etc.

3.5.1 Apple AppStore Workflow



Apple AppStore "App Status" is detailed in the following reference:

https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Chapters/ChangingAppStatus.html

Yellow:

Prepare for Upload	Waiting For Upload
Upload Received	Waiting For Review
In Review	Pending Contract
Waiting For Export Compliance	Pending Developer Release
Processing for App Store	Pending Apple Release

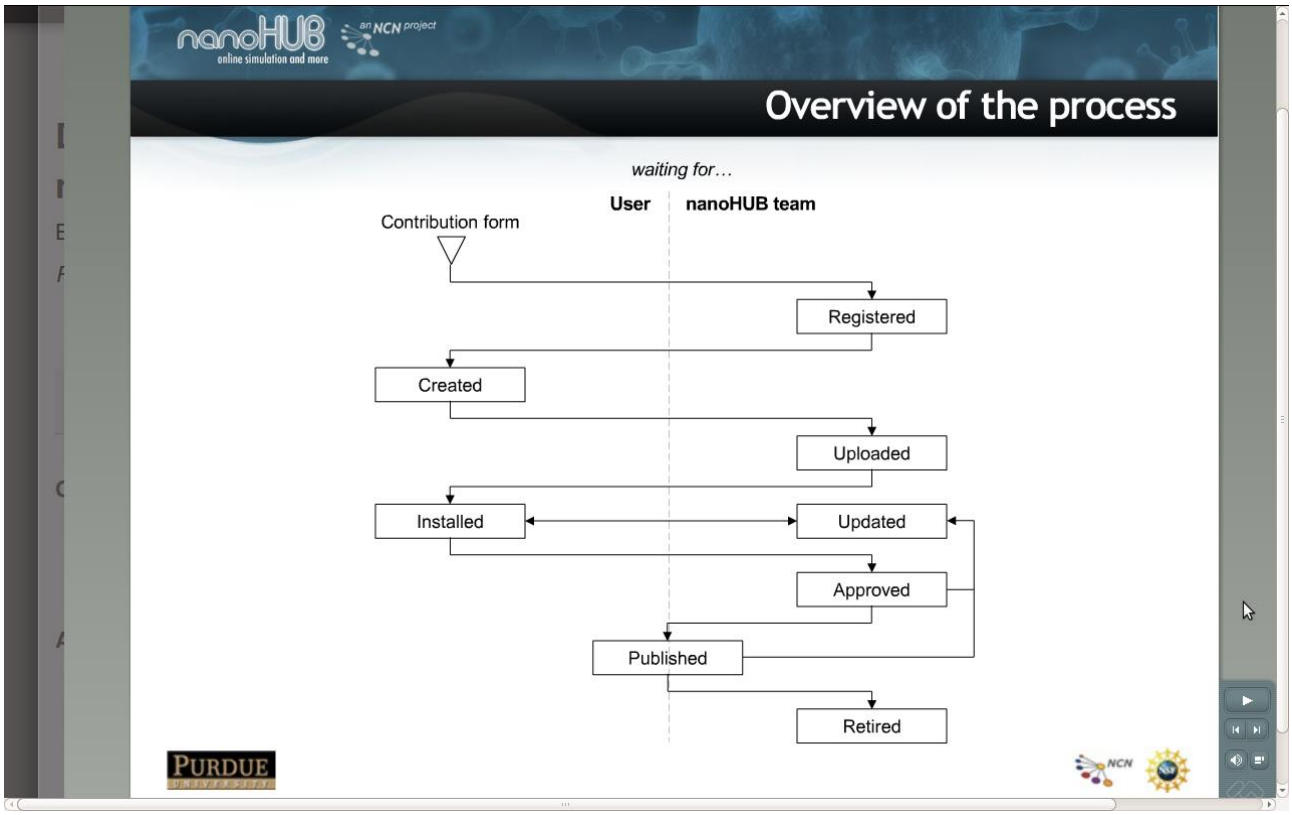
Green:

Ready for Sale

Red:

Rejected	Metadata Rejected
Removed From Sale	Developer Rejected
Developer Removed From Sale	Invalid Binary
Missing Screenshot	

3.5.2 nanoHUB Tool Contribution Workflow



The nanoHUB project has less number of statuses, 8 of them: Registered, Created, Uploaded, Installed, Updated, (Developer) Approved, Published, Retired. The "Installed" status is needed since nanoHUB aims to support software as a service (SaaS), so it provides a "project area" to a tool developer to upload, edit, and test tool software.

3.5.3 Brown Dog Tool Workflow

Compared with nanoHUB's workflow, the Brown Dog Tool Store project does not seem to need to provide "project areas", so there seems no need for an "Installed" status.

The Tool Store can have the following statuses:

(a user successfully creates and submits a tool contribution form on the web)

→ Registered

→ (the user successfully uploads the corresponding bundle to the Tool Store)

→ Uploaded

→ (the user reviews the information in the Tool Store, making sure the content, look and downloading are as expected, then clicks a "User Approve" button)

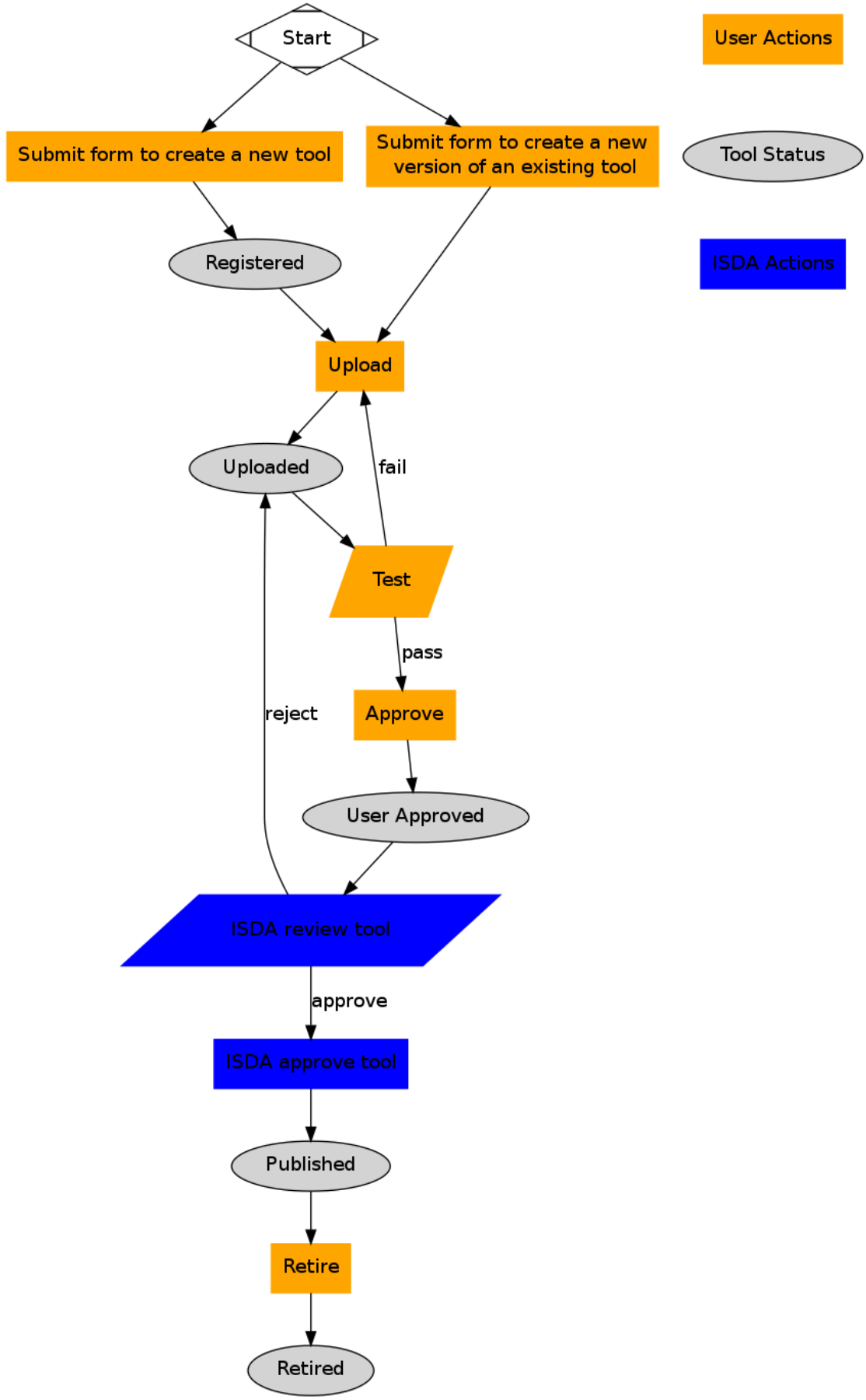
→ User Approved

→ (Tool Store administrator/ISDA team performs sanity testings/reviews, then either rejects it, or publishes it)

→ Published

→ (the user can either add a new version, going through the "Uploaded" → "User Approved" → "Published" path with the new version again, or retire an existing version)

→ Retired (the version of the tool with this status will not appear in the Tool Store for other users)



3.5.4 A Tool Is a Bundle/Archive

A tool might need to contain multiple files, so may require uploading a bundle (in Apple AppStore term), i.e., an archive, likely a compressed tar ball/zip file, containing:

the tool file or files themselves, supporting docs (such as installation and uninstallation instructions, user guide), icons, screenshots, license, etc.

We need to decide on its format.