Day 2, PM : Carole Goble, Alice Allen, Ashley Sands, Patricia Cruse

Edited Use cases

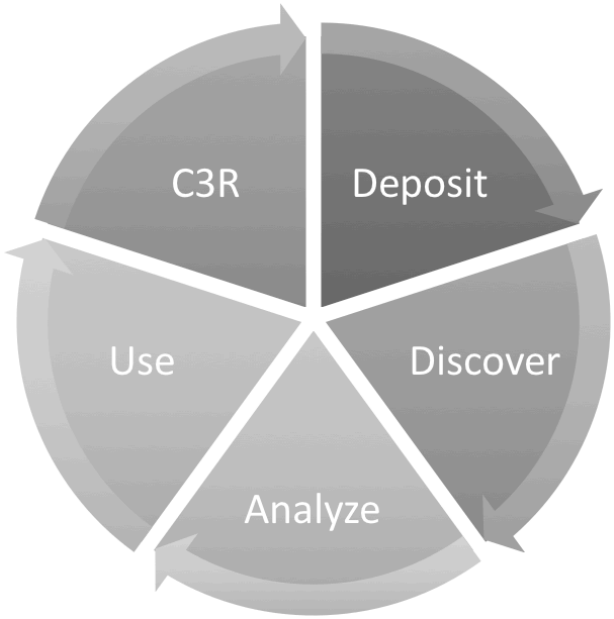
Created 20160415 (use cases group pulled from here 3pm 20160516)
CodeMeta meeting in Portland
This document is the electronic version of the “whiteboarding” we did in the workshop

	Stakeholders/Roles (Users of the metadata)								
Action* (on the software)	Indexers	Developer	Repository Mgr.	Researcher	Funder	Publisher	Citation Mgr	Library / Curator	Utility Infrastructure
Deposit**	x	x	x	x				x	
Discover: 3, 4	x	x	x	x	x	x	x	x	
Analyse: 1, 2, 6, 9, 11		x		x			x		
Use		x		x					
C3R: 4, 5, 6	x	x	x	x	x	x	x	x	

TO DO:
*add subpoints to each action
**add concepts (by #) under stakeholder for each relevant action sub-point

Notes: September 12, 2016 (meeting with Carole and Alice)
IN the table, the top row is of Stakeholder ROLES (users of the metadata)
And the first column is of ACTIONS on the software.
SO, someone who wants to DEPOSIT software will have the role of an indexer, developer, repo mgr, library/curator (as an example of how to read the table).
DEPOSIT includes push (action from contributor, which could be developer, repo manager, researcher, or indexer) or pull (action by indexer, for example)

Day 2, PM : Carole Goble, Alice Allen, Ashley Sands, Patricia Cruse



Continued debate: Unclear whether deposit and discover should be separate entities. Right now, those are separate because they are distinctly different stakeholders (the creator, the reuser). However, in the other three, we do not separate the stakeholders (ie want to enable analysis, and analyze are both in “analyze”). So, that’s the inconsistent thing right now, however we like the way this figure is developing in terms of branding codemeta.

1. Identify software functionality
    a. Overall
    b. Module level
    c. Function level
    d. Identify functional equivalency with other {modules, functions, etc)
2. Understand dependencies
    a. Identify Installation Dependencies (User)
    b. Identify Development/Build Dependencies (Developer)
    c. Use dependencies to enable transitive credit
    d. Automate build processes
    e. Identify dependencies to assign (aspects of? partial?) credit
    f. Understand role of software in an application (even if we cannot access or install)
3. Enable Discovery (find it)
    a. Support discovery for reuse
    b. Discover tools that can perform a specific function
        i. Enable comparisons of tool functionality

Day 2, PM : Carole Goble, Alice Allen, Ashley Sands, Patricia Cruse

   c. Classify and find tools by domain, and other controlled vocabularies
   d. Locate tools that were used for particular applications/results
   e. Discover location of software
4. Enable citation, credit, compliance, and evaluation
   a. Identify roles of software contributors
      i. Authors
      ii. Metadata authors
      iii. Curators
      iv. Other roles (see publisher's credit ontology)
   b. Document compliance with funders policies
   c. Identify funder of software
   d. Citation
      i. Cite software product generally
      ii. Cite a specific software version/release
      iii. Cite a software paper describing the software
      iv. Provide sufficient info for formal citation in the reference list (authors, title, repository, publication year)
      v. Link to a landing page with more detailed information about the software
      vi. Link to the source code
   e. Count the citations to a given version of software
      i. For acknowledgement/credit
      ii. As a quality metric for discovery
   f. Aggregate by person, by infrastructure, etc.
5. Enable Interoperability
   a. Describe platform, language constraints
   b. Describe input-output formats
   c. Describe input-output semantics
   d. Enable pipeline building
   e. Describe constraints on parameters
   f. Identify license obligations
6. Describe provenance information
   a. Execution parameters, limitations / assumptions
7. Enable transparency and reproducibility (via dependencies and source linkages)
8. Communicate permissions / License terms
9. Identify software status (See http://www.repostatus.org)
   a. Build status
   b. Maintenance status
   c. Understand intention of software publication (compliance, reuse, etc.)
10. Motivate archiving of software
11. Evaluate software quality

Stakeholder list pulled from SCWG:
https://github.com/force11/force11-scwg/blob/master/sc-principles/software-citation-principles.tex

Stakeholders: Digital Repositories, Publishers, Information Science, Indexers, Software Developers,

datacite/crossref/orcid -- utility providers

`Researcher'' includes both academic researchers (e.g., postdoc, tenure-track faculty member) and research software engineers. \item

``Publisher'' includes both traditional publishers that publish text and\slash or software papers as well as archives such as Zenodo that directly publish software. \item

``Funder'' is a group that funds software or work using software. \item

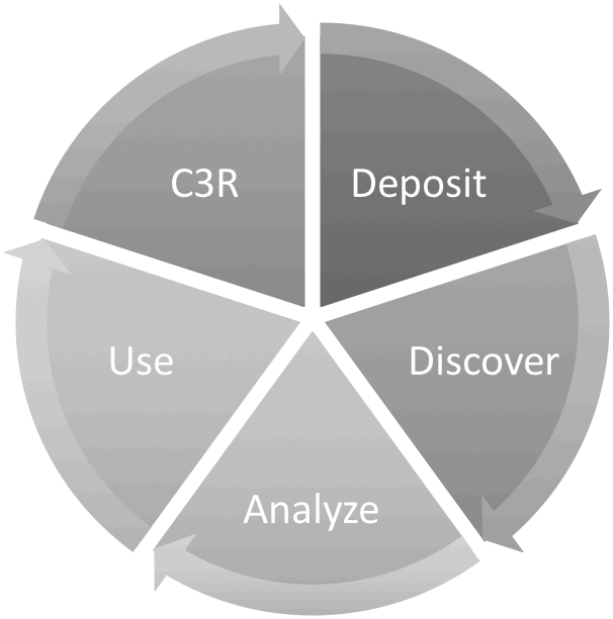
``Indexer'' examples include Scopus, Web of Science, Google Scholar, and Microsoft Academic Search. \item `

`Domain group\slash library\slash archive'' includes the Astronomy Source Code Library (ASCL)~\cite{ascl}, bioCADDIE~\cite{bioCADDIE}, Computational Infrastructure for Geodynamics (CIG)~\cite{CIG}, libraries, institutional archives, etc. \item

``Repository'' refers to public software repositories such as GitHub, Netlib, Comprehensive R Archive Network (CRAN), and institutional repositories. \item `

``Citation manager`` refers to people and organizations that create scholarly reference management software and websites including Zotero, Mendeley, EndNote, RefWorks, BibDesk, etc., that manage citation information and semi-automatically insert those citations into research products.

Day 2, PM : Carole Goble, Alice Allen, Ashley Sands, Patricia Cruse



Day 2, PM : Carole Goble, Alice Allen, Ashley Sands, Patricia Cruse