TechWPaper Documentation

Release v0.1

Wes Turner

October 18, 2014

1 Report 1

2 *Presentation* 3
2.1 Self-Directed Learning with Online Resources 3
2.2 Challenges 5
2.3 Opportunities 12
2.4 Strategies 14
2.5 License 39

3 Process 41

4 *Glossary* 43
4.1 Glossary 43

5 *Index* 57

6 *Search* 59

7 License 61

CHAPTER 1

Report

CHAPTER 2

Presentation

2.1 Self-Directed Learning with Online Resources

An independent study of challenges, opportunities and strategies for encouraging feedback between tools and resources in online learning systems

`aquariums and cave diving`

Note: might as well be a request for proposal

2.1.1 Objectives

Self Directed Learning

Autodidactism: Self-Directed Learning

- Time
- Skills
- Discipline
- Rhetoric
- *Resources*

And tools that *augment* learning in application to:

- Theory
- Knowledge
- Wisdom

Aquariums and Cave Diving

```
Aquariums and Cave Diving
```

- Jacques Cousteau
- Edutainment/Infotainment
- Documentary :: Reality

- Information Chunking
- Market Study Confidence
- Attention Retention

Undergradate STEM Graduates

"We need one million more *STEM* trained:"

- Qualified, Knowledgeable Consumers
- Efficient, Sustainable, Creative Producers
- Data Science Influencers

2.1.2 Background

Learning Resource Timeline

```
          Printing   Newspaper
           Press      Telegraph
                        Magazines
                         Radio
                           Television
                            Internet: URLs
                              Wireless Internet
                              Mobile Internet
                              Tablets
                        Wireless
Stone Papyrus        Copper   Fiber
BC     CE    1800     1900     2000     2100
```

[TODO:cite][TODO:correct]

Knowledge Economy Roles

Producer

- Creating
- Synthesizing
- Distilling
- Adding Value

Consumer

- "Smart Consumer"
- Searching for Value

Influencer

- Continually learning the market

  When are the dances?

Yearly Knowledge/Data Graph

```
Stream/stacked graph by source with rough yearly data transfer
```

[TODO:cite]

2.2 Challenges

- Room to Grow
- Resource Efficacy
- Cost/Return Curve
- Creativity & Process Control
- Tool Frustration -> Resistance to Change
- Taking a step back

Reaching People with Busy Schedules

- Data
  - Charts
  - Analyses
- Opportunities
- URLS
  - Bookmarkable
  - Shareable
  - Referenceable
  - Archivable

2.2.1 Resource Constraints

Learning Materials

Learning *Resources*

Course Artifacts and Components

> "I want to download all of this as a zip file"

- Syllabus
- Lectures
- Slides
- Quizzes
- Assignments / *Activities*
- Expected Assignment Durations

- Documents: *PDF*, *HTML*
- Audio
- Photo
- Video
- Books
- Web Pages
- Datasets
- *Linked Data*
- Knowledge Triple Repositories
- *URLs*
- Notes
- Questions
- Answers

Resources: Course Lifecycle

> "Why do we do it this way?"

Answer

We used to ___________, but __________,
so we tried __________, but __________,
and now ________________________________,
so what we're looking to do is

____

on the *web*

with *revision controls*

```
Topics -> Requirements -> Research -> Expertise (Reivew) ->
   Chunking Wisdom -> Formatting -> Publishing

   Publishing:
      PDF
      Books
      Slides
      Tapes
      CDs
      WebPages
      Knowledge Repositories
      Databases
```

2.2.2 Employer Acceptance

Verification

2.2.3 Incentivization

Human Motivation

Why?

2.2.4 Learning Assessments

Self Evaluation

Standard-Normal Testing

Voting Choice Theory

Essay Qualification & Quantification

2.2.5 Sharing Resources

Creating and Updating Resources

What does it mean to produce a *hyperlinked* document?

- Why HTTP?
- Why HTML?
- Why Javascript?

Information Access Optimizations

- *Table of Contents*
- *Index*: Terminological Lexicon
- *Glossary*: Compendium of Definitions
- *Search*

Bibliographic Citations

Maintaining structured links to reference and documentation sources

- Document title
- Page Number
- #URL Fragment
- Attachment Revisions

Sharing Data Resources

> *Neat*, **Awesome** Study, but
>
> - What is this *named*
> - How easily can our tools share data?
> - Where can I download the checksums?
> - How do I download it?
> - What do you want to do with it?
> - What concepts and keywords does it reference?
> - Which part of this should I translate?
>
> "One of them is 'Naming Things"' – TODO

Resources: Bookmarks

> - Bookmarks are great
> - Bookmarks are resources
> - I need to consolidate my bookmarks
> - I need to manage my bookmarks
> - I need a bookmark manager
> - I want a bookmark management system
> - I want to upload my bookmarks
> - And search them
> - And paste them to you
> - At opportune moments
>
> Check this out.

Document Capturing Process

> Wait? What? You want to scan this? Awesome.

Artifacts

- Why tag things?
- Why name them?
- Why label them?

Workflow: Sharing Documents as Email Attachments

I want to email a document that would be better if we were to revise collaboratively.

Why would it be inefficient to email variously named and occasionally versioned versions of a document resource?

When do we file this?

Designing URIs for Resource Collections

> What page number was that? If this was a web page, I could just __________?

- Which page is it on?
- What is the *URL* ?
- Why doesn't this *PDF Bookmark* include the document title?
- This page does not include content mentioning these keywords.
- Can I link to a document subsection with a *URL Fragment*?
- How should I organize course files for a zip archive?
- How should I organize course files for packaging?
- How do I watch/listen to course lectures on a phone, tablet, TV, mac, pc?
- How can I adjust the font?
- Where should the *presentation* mix with the *Content*?
- "Data not available"
- How do I search based on *metadata*?
  - Can I facet the search query with certain criteria?
- Can I share this link?

Versioning Project Resources

Managing a **Project** as a *Resource* with versioned links to additional *Resources*

Comparing wikis and version control

Wiki

- Revision Trail: "History"
- Talk Page: Commentary, Feedback

Version Control

- Revision Trail: "Commit Log"

2.2.6 Interface Usability and Accessibility

Interfaces for Online Learning

> "Really what I want is..."

- Focused learning activities
- search
- animated
- four-dimensional
- through my phone/tablet

Managing and encouraging feedback at scale

Designing flexibly linked data models to support cool *UIs*.

Interface Criteria

- Where is the table of contents?
- What do `Page Up` and `Page Down` do?
- What about `Tab`?
- Why do I have to **click** next and previous?
- Can I `Ctrl-F` to search within the resource?
- Can I tab through these?
- What can I link to?
- Can I link to a URL fragment in a new tab?
- Can I scroll with the mouse wheel?

Interface Questions

- Can I download a list of my assignment due dates?
  - To my iCal
  - As a list of tasks with duration
  - As an RSS feed
- What does it mean that a course is worth N credit hours?
  - How many hours a week is that?

Separating Content and Presentation

> Which page number is it on my device?

```
 ____________________________
|        ___________________|
|       |    _______________|
|       |    |    __________|
|       |    |    |    _____|
|       |    |    |    |    |
|_______|____|____|____|____|
```

I want to review this course on my

- Laptop
- TV
- Tablet
- Phone
- Screen Size
- Bandwidth

- External Media
  - USB Drive
  - SD Card
  - MicroSD Card

Web Standards

Cascading Style Sheets

- Text Reflow
- Fluid Layouts

2.2.7 Privacy and Security

Privacy

> "How do we verify this curriculum vitae?"

Status Quo

- Limited Records
- Access Logs
- Retention Policies

Standards

- FOAF (Friend of a Friend)
- OAUTH API Controls
- OpenID

Note: *Health Photo Journal* Balancing Act

- Personal student health information?
- Privacy Controls for remotely hosted resources
- Difficulties of data scrubbing

Security

> Which email is this linked to?

Assurances Necessary for Measuring Learning

- Confidentiality
- Integrity
- Availability

2.3 Opportunities

2.3.1 Direct Returns

Teacher Salaries

2.3.2 Indirect Returns / Externalities

Network Effects

Goodwill

Knowledge Economy

STEM Theory, Process, and Knowledge

Learning STEM *Theory*, *Process*, and *Knowledge* through applied data science.

- **Linking things together**
- Feedback and Collaboration
- Communication Channels
- Q&A

Goal: Maximize Collaborative Output

Criteria for evaluating collaborative learning products

Network Effects n^2

- Jointly Authored Pages
- Information Access
- Streaming

[TODO:cite]

Generalizing Educational Momentum

- Stratified skills objectives
- Measuring for
  - *Ability*
  - *Aptitude*
  - *Proficiency*
- Serial Order Effect

Knowledge Graph

Ontologically related fields

Graphs of *Knowledge*, *Skills*, *Competencies*

Overlaid onto a *knowledge graph* of *resources* and their *artifacts*

Subgraphs

- People
- Citations
- *Learning Objects*
- *Curriculum Sequences*
- Degree Requirements
- Progress Flows

Media

- *Documents*
- Photos
- Recordings
- Videos
- *URL*
- *Datasets*

Linking Between Things with URLs and Metadata

- Bibliography Lists: *BibTeX*, *APIs*
- *Document Repositories*: Document Identifiers
- Document Authoring: naming URLs
- *Bookmarks*: Adding tags and attributes
- *Learning Objects*
- Utilizing *Web Standards*

Tools: Pushing for Better Calculators

> "Where do I set the calculator?"

Why is it that we would be interested in pushing for:

- Better Personal Calculators
- Computers, Notebooks, and Tablets

Publishing, Storage, and Presentation

> Where should I put my slides?

Answer

- The Shared Drive
- Your USB drive
- Your Cloud Drive
- Cloud Presentation Services
  - Commenting

2.4 Strategies

> "How do we make this better?"
>
> "There has to be a better way of doing this"

- I need to write this down
- Can I just lay it all out?
- Can we list it in a table?
- Can I add columns?
- Which columns can I add?
- Which columns are already there?
- Which columns should I add?
- What did they call a `Widget`?

2.4.1 Channels

Publishing Static Documents

Video

- Growth Metrics
- Bandwidth minimzation
- Broadband availability
- Replicated hosting
- Metadata API
- Subtitles API
- Referencability

[TODO:cite]

Q&A

> "But they are sharing the answers!"

Asking for answers.

> "You want to answer questions?"

Q&A Web Sites

- http://stackoverflow.com
- http://quora.com
- *askbot*

Social Media

- Whiteboarding
- Games with friends
- Video, Audio, Text Chat
- Linked Personal Profiles

2.4.2 Implementations

OpenCourseWare

MOOC: OpenCourseware, Coursera, EdX

MOOCs: Massive Open Online Course

Foregoing the unused 80% of *LMS* and *LCMS* features for *scalability*

- http://opencourseware.mit.edu
- http://coursera.edu
- http://edx.edu
- http://udacity TODO

MOOC Online Course Features

- Web Hosting
- Video Hosting
- Linking with readings
- Optional Books
- Relevant papers
- Device/Browser Support: Mac, PC, Android, iOS

Online Course Experience

> I was listening to Andrew Ng's 2012 Machine Learning course and glancing at the subtitles streaming over a wireless laptop, and thoroughly appreciated the ability to rewind.

[TODO:cite]

Software Programming: CodeCademy

Math-based web game design

University Online Offerings

# TODO -> LCMS

Learning Dashboard

```
Annotated countour/heat maps

* Activity/Objectives
* Student/Objectives
```

Charting Competencies

- Signing Authority
- *Activity Metrics*

[TODO:cite]

2.4.3 Authoring Tools

TODO

Authoring Tool Features

Standard Output Formats

- *HTML*
  - *linking*
  - *bookmarks*
  - *metadata*
- *PDF*
  - *documents*
- *SCORM XML*
- *Zip*
  - *archive*

- Mobile Support
  - What is the page number on my device?
  - How deep are these links?

Format Support

- *Artifacts*
- *Document Authoring Tools*
- *Web Standards*
- *Browser* standards benchmarks
- *SCORM* Sequences -> *TinCan*
- *TinCan API*
- *Activity Metrics*

Standards: Web

> What is the *link*?

Links, URLs, and URIs

Describing *resources* with *URIs*

```
http://   example  /pages/page  ?query  # contents
scheme    hostname path         query   fragment
```

```
http://example.com
```

HTTP

HTTP request response

HTML

HTML * Storage * Archival * Versioning

Metadata: Searching & Indexing

- RDFa
- Microdata
- GIFT

Web Video

TODO:cite

Standards: Education

Means for *linking* between *learning objects*

SCORM *TinCan API*

SCORM

SCORM

- Glossary Vocabulary Model

TinCan API

TinCan API *Activity Metrics*

- Verbs
- Scores
- Times

TODO:cite USA ADL

Standards: Markup Languages

Markup Languages

- *SGML*
- *XML*
- *HTML*
- *HTML5*

```
Standards Graph

SGML -> HTML
SGML -> XML
SGML -> SCORM
XML  -> SCORM
HTML -> PDF
RST -> PDF
RST -> Latex
RST -> JSON
```

TODO: dot rendering

Standards: TinCan API

Recording *Activity Metrics* into a *LRS*.

Web *API*

- *JSON*
- *XML*

JSON

```
{
  TODO
 }
```

TODO:cite

Signing Authority

Note: The subjectivity of activity pings

SCORM in HTML5: TinCan Activities

SCORM containers

- *Web Hooks*
- *Activity Metrics*
- *TinCan API*

TinCan API:

```
SUBJECT_X   ACTION_VERB    RESOURCE       SCORE    DURATION
```

Lightweight Markup Languages

Simple Plaintext *Markup Language* for generating Multiple Output Formats:

- Books
- Web Sites
- Documentation Sets

Features

- Bold, Italic
- Inter/Intra- document linking
- Footnotes
- Citations
- Figures
- Tool Chain
  - Language Portability
  - Web Editor Support
  - Syntax Feedback Loop

Transformed between various *Markup Languages* at *build/compile* time:

- *ReStructuredText*
- *HTML*

- *PDF*
- *EPUB*
- *TXT*

Stored in a changeset of a *DVCS* repository.

Organizing a Sphinx Documentation Project

Folder of files with:

- an index manifest
- intra-document links
- citations
- glossary terms

The root index.html is expected to branch (and link) into the *document graph*.

To write a paper, which may later be a book:

- *HTML*
- *PDF*
- *EPUB*
- *JSON*

Examples:

- http://docs.python.org
- http://hgbook TODO
- http://python-guide.org
- http://scipy-lectures.github.com
- http://readthedocs.org

Knowledge Resource Criteria

- Does it have *permalinks*?
- Is it possible to reference *media repository* and external URL resources in an *authored SCORM* course?
- What is the best way to represent a sequence of graph constraints as rules? (*RIF*?)
- Can I *bookmark* this part?

Authoring: Sequencing Learning Objects

For & Against Task/Skill Dependency Models

Cost / Complexity / Waste

- Organic Model -> Choose your own adventure
- Cost/Path Routing: *Sequence Optimization* Wisdom

UI / UX

- Browser
- Game Console
- PC Game
- Virtual Learning Environments
- MOOCs

2.4.4 Learning Sequences and Curricula

Sequencing Exercise: Pen and Paper

1. Generate: Brainstorm 50-100 sheets of paper
2. Reduce:
3. Clarify:
4. Organize:
5. Evaluate:
6. Build Consensus

TODO:cite: Process: Six Patterns of Collaboration

Curriculum Interfaces

> would be great if I could schedule sliding windows of content sequences with expected exercise durations.

A Curriculum Sequence

1. **Read A1 Reading** Consume *document resource*
2. Watch **A1 Video: Lecture 1** Consume *resource*
3. Answer **A1 Quiz Questions** Check for comprehension.
4. **Apply A1 Project** Apply and synthesize.

Comprehension testing, *Peer-reviewed annotations*

Sequencing Tagged Resources

> “Twist until it feels correct.”

Learning Objects and *Learning Resources* into *sequences*.

Paths of *Learning Objects*

Paths build upon themselves

Encoding domain wisdom into *sequences*

Authoring: Generating Curricula Activity Graphs

As knowledge graph traversals that fit degree and resource constraints.

Sequenced graph traversals with boundary-contextual transitions

Process Component Competency Graph

Curriculum/Degree Sequence Requirements

> What is the path to the file?

Describing learning areas

- Goals
- Modules
- Assessments
- Prerequisites as implied paths/sequences

Describing **Learning Momentum** with stratified **skills objectives**

Measuring for *Ability*, *Proficiency*, and *Aptitude*

URI-linked resources and reasonable concepts in a linked open data cloud.

Modular interface specifications with prerequisite knowledge linked as part of a traversal through a greater knowledge graph.

Graphs and subgraphs of *Learning Resources* with *URIs* according to domain-relevant *ontologies*.

Reviewing Curriculum Sequences

> Is there a book?

Sharing Revisions and Annotations

- Revisions: Corrections, Extensions
- Notes/Comments
- Feature presence and fitness ratings
- Comparing Citations

Note: *ReStructuredText* + *GIFT* + *OEMBED*

Authoring: Curricula Management

So we have standardized curricula which can be delivered through multiple channels.

- How do I add/modify this?
- How do we share challenges & opportunities?
- How can we compare similarities and differences?

Distributed Revision Control

- Peer Revision Control

- Local Forks, Branches
- Pull Requests, Patches

Learning Metrics

> "Is it the course or the student?"

- Curriculum Goal Sets
- *Optmization*
- Conceptual Routing Algorithms

2.4.5 Software R&D and Learning

Note: clearly, what is being learned is that analysis of structured data interchange in regards to optimizing for outcomes

Learning Comes from a diverse background

Note: it would follow that software development learning best practices are inherent and implicit to best practices for software development process control

Process Wisdom

the software is the person is not the curriculum

Project Based Learning

Note: So, okay, well these assignments look great but we're going to work on developing projects so that we can focus on specialization and synthesis of TODO

- How should we measure success?
- Which feature scores correlate with success?
- Learning Scientific, Creative and Collaborative Processes

Augmenting Learning

> "Here's a library, knock yourself out"

Simplest, least complex approach Accelerating and Augmenting Learning

Englebart 1964

Standard Project Documents

- README: Project Goals, Objectives, URLs
- TODO: Task Lists
- CHANGELOG: Project Change Log

Revision Control

- Commit changes to files
- Commit Log
- Manifest
- Link to TODO/Issue/Ticket item numbers
  - `working on #123:  this, this, and this`
  - `working on #123`
  - `task:course/project/n: this, this, and this``

Open Source Software Development Community

Healthy Online Communities

- Documentation: Sphinx/ReStructuredText
- Issue and resolution tracking
- Proposal Review Workflow
- Learning Community
  - Wikis, Blogs
- Books: Paper, Digital, Web
- Distributed Version Control

[TODO:cite]

Version Control Workflow Development

```
1. edit, edit, commit
2. edit, commit
3. todo, edit, commit
4. todo, edit, test, commit
5. todo, test, edit, test, commit
6. todo, test, edit, test, commit, tag

7. todo, branch, test, edit, test, commit, tag, push
8. todo, branch, test, edit, test, commit, tag, send patch
```

Collaborative Review

TODO

Reviewboard

- Patches & Comments
- Review & Approval Workflows
- Changesets as versioned patch queues

Compile a Course

- Choose *Sequence* Representation *Standard*
- Check/Verify Content Links
- Flatten Metadata to Bibliography
- Link References, Footnotes, Citations, Glossary Terms
- Generate Sequence Manifest
- Link *Activities* to a *LRS*
- Generate Indices
- Template Packaging
- Schedule Course Dates
  - Calendaring

Continuous Regression Testing for Learning

- Testing Linked, *namespaced components*
- Test Cases
- Build & Test Metrics
- Test *Coverage*

Sequences

- Linear
- Goal Objective
- Path-Based Routing
- Serious Games

Spaced Repetitions

Mnemosyne

`spaced repetition chart`

Optimization: Machine Learning for Sequence Optimization

Optimizing yields from *activity graphs* and *Activity Metrics*

- Variance
- Confidence
- A/B Split Testing
- Cost/Flow algorithms
- Rotation, Mutation, Mutability

- Logarithmic Logistic Crossover
- Mutual Information Coefficient
- Learning Perceptron Weights
- Perceptron Graph Serialization
- Vertex Messaging
- Event Queues

Optimization: A/B Testing

This document modification yielded this increase/decrease.

```
TODO: Content Branch Crossover
```

Optimization: Cost/Flow

Cost flow calculation of a path toward subgraph objectives.

```
TODO: a river
```

Optimization: Receiver Operating Characteristics

- Learning Process Knowledge
- Identifying Productive Personal Behaviors
- Identifying Productive Organizational Behaviors

Note: similar to optimizing for conversions

```
TODO: ROC quadrants
```

```
TODO: ROC Curve
```

Optimization: Streamgraph Conversion Funnel

```
TODO:chart
            STEM Specialization

General Education    Undergrad   Grad  Doctorate

stacked/stream graph
```

Searching and Indexing

- Index
- Cluster
- Archive

- Share
- Normalizing
- Eventual Consistency
- Document Checksums
- Computed Attribute Indexing

  which part do I copy to the search service?

Namespacing

TODO

- Namespace:
- URL:
- Query:

Finding and Disambiguating Concept & Instance Identifiers

> How do our fields link together? What do they call a _____ ?

- *Concept*: `Bookmark`
- *Instance*: `bookmarks/{id}`:

  ```
  { 'ID': ID,
    'isA': Bookmark,
    'url':URL,
    'title': TITLE }
  ```

- *User Instance*: `bookmarks/{username}/{id}`

Note: Nodes and edges with activities whereupon potentiation potentiates with behaviors determined by namespaced attributes

Collaboratively linked ontologies for augmenting sequence authors

Cognitive Ontological Knowledge Graph

- Concept Trees
- Concept Vines
- Namespace Prefixes
- Typed Links/Edges
- *RDF* / *OWL* / *SPARQL* / *JSON*

Linked Open Data: Wikipedia and DBPedia

Value

Shared, Multi-Lingual Concept Identifiers

- Translate Terminological References

- http://en.wikipedia.org/wiki/Wikipedia
- http://jp.wikipedia.org/wiki/Wikipedia

```
TODO: SPARQL SELECT query
```

Note: Wikipedia is written in *MediaWiki* syntax

2.4.6 STEM Labs

STEM Labs: Laboratory Based Learning

- Learning process knowledge
- Team Collaboration
- Experimental Design
    - Null Hypothesis testing
    - Drawing conclusions about correlation and causation

STEM Labs: Data Science Laboratory

Skills

- Data Science
- Sequence Modeling
- Software Development
- Process Modeling
- Process Controls

"Development Lab" of virtual *services* and *resources*, per-course and/or per-student.

- Objectives
- Processes
- Tools
- Data

```
               Processes
Theory      Procedures     Tools          Knowledge

Books       Repeatability  Spreadsheet    Wisdom
Libraries                  Notebook       Metaheuristics
```

```
TODO
```

STEM Labs: Services

- Q&A
- Apps Accounts

- Web Hosting
- Repository Hosting
- Project Hosting
  - *GitLib*
  - *Fossil*
  - *Trac*
- *Cloud* Service

STEM Labs: Student API

- Name
- Courses
- Skills
- *Activity Metrics*
  - Tasks / Activities / Implemented Services
  - Completions / Activity Logs
  - *Badges*
- Implemented Services
- Grid Resources
  - Capabilities *Access Controls*
  - Virtual Server *URLs*
- *Resources*
  - *Document resources*
  - Project *URLs*
  - Structured Bibliographies
  - Portfolio / Curriculum Vitae

STEM Labs: Student Assignment API

> What is a continuous learning ensemble?

Activity Objectives:

- Complete an *ipython notebook* worksheet
- Run functional tests
- Store changes in a *repository*
- Integrate the worksheet with an API interface for grading
  1. Create a POST *HTTP Request* to an *API*.
  2. *API* returns *HTTP Response* with evaluation

  "Make yours like mine."

```
Code Repository      Code Repository
Local Workspace      Activity Tests
Books
Lectures
API     <--------->  Build Server

Student              Course

Testing Components of a Learning Ensemble
```

Note: Increasing test coverage.

Repository Event Hooks

```
def on_commit(repository):
    # check syntax
    # strip answers
```

URL Schema:

```
http://hostname.edu/{coursename}/{courseunit}/{taskname}#{task_link}
```

Grading Criteria:

- Does it match tolerances?
- Is it repeatable?

STEM Labs: Setup Python Application Framework

Objective: Continuously test submitted problem/test solutions with a controlled build server that *requests* answers from a student-implemented *API*.

- Application Framework Scaffold/Skeleton
- With a build script
- And documented extension points

Relevant Courses:

- Software Development
- Web Development
- Agile Project Mangement

STEM Labs: Python Tools

Tools for Developing *STEM Python* Applications

- *ipython* notebook
  - visual charting
  - repeatable processes
  - procedural spreadsheets
  - testable spreadsheets

- Science Libraries
  - *scipy*
  - *numpy*
  - *pandas*
  - *scikits*
  - *statsmodels*
- Version Control
  - *Mercurial*
  - *Bazaar*
- Issue Tracking
  - *Trac*
- *Python(X,Y)*

TODO:cite

STEM Labs: Cloud Services

Primary Services

- Deploy Applications
- Perform Load Balancing
- Allocate Identity Service Credentials
- Allocate Grid/Cloud/Stack Resources
- Alloccate IP Adresses
- Allocate Service API Keys
- Allocate Grid Credentials

Configuration Management

- Bootstrap Server
- Freeze/Save/Snapshot Server
- Generate new home directories

STEM Labs: Open Learning Stacks

- *OpenCourseWare*
- *Open Stack*
- *Open Source*
- *Open Access*
- *Linked Open Data*

STEM Labs: Media Repository

Storing versioned media *resources*

Resource Repository Standards

- OAI
- REST API
- *Metadata*

Build a Bookstore API

- Citations, References
- Books
- Films

API Lookup

- Bookstores: AMZN, APPL, GOOG
- Social Web: GoodReads, LibraryThing
- WorldCat API

2.4.7 Identifying Tools Gaps as Opportunities

# TODO -> Feature Requests

Communicating Feature Requests

Sample Templated Use Case:

```
As a _________ <user>,
when I ______________,
it ________________,
but it/I ________ <would/should/could>
_________________,
which would save __________ <amount of time>.
```

Note: Eliciting Value Criteria Matrices

Achieving Feature Decision Consensus

- Feature Value Elicitation
- Solution Evaluation
- How important is this feature to you? 1-10
- How many _____ <noun> is this feature worth?
- How many ______ <noun-plural> do we have?

Learning Content Management Systems

Examples of *Learning Content Management Systems*

- Sakai
- Moodle
- Blackboard

Someone has a LMS/LRS *Feature Matrix*

Rubrics and Feature Matrices

> Grading *Rubric* :: *Feature Matrix*

Document Publishing Feature Comparison Matrix

A *feature matrix* for comparing *components* of a *document publshing* system

- *Authoring Tools*
- *Learning Content Management Systems*

```
Feature                    Choice-0    Choice-1    Choice-n
Category1.
   .Feature A
Interfaces
   .HTTP
   .HTML
   .Javascript
   .JSON
   .XML
   .CSV
Hosting
   .Standard Web Hosting
   .Local Application
   .Hosted Application
Code
   .Open Source
   .Proprietary
Support
   .Agreement
   .Yearly
   .Monthly
```

Gap: Freemind to ReStructuredText

Objective: Transform between a *ReStructuredText* document set and a *Freemind* mind map

Value: Generate document outlines from mindmaps

```
MindMap to Paper

If I am creating a linearly sequenced book, I am essentially creating a
depth first search/walk of an ordered graph of document sections with
hopefully compile-time checking for links and references
hyperlinked within and between documents with markup.
```

Gap: sphinxcontrib-courses

- Language to describe questions, answers, and sources
- Roles and directives for referencing Q&A with learning object resources
- Check and cite links to learning resources
- *Metadata attributes* with *URIs*

Docutils

- Roles
- Directives
- Publishers

Language to describe the course curricula

- Processes
- Content
- Questions
- Answers

Q&A Authoring

Syntax for expressing Questions and Answers in *ReStructuredText*

Re-*sequencable* tests for comprehension:

```
1,2,3,4 ; 4,3,2,1 ; 2,1,3,4
```

Similar to web commenting services, but with questions and various categories of answers

- Q&A *Authoring Markup Language*

  > Objective: Embed tests for reading & skills comprehension intro a content authoring markup.

- *Link* Q&A *activities* to a *TinCan LRS* and an *LMS* with a *Learning Dashboard* for identifying opportunities to improve.

Allow for question/label randomization and *optmization*

- If 85% answer correctly with sequence 0 and wording 1
- If 95% answer correctly with sequence 1 and wording 0

Note: Audience testing

> "But they are sharing the answers!"

Branch Commit Preprocessing Hooks

TODO:cite

Gap: Per-Course Sidewiki

- Per-course sidewiki
- with scholarly overlays
- and an answers tab, for teachers

Process: Citations

Tools: Managing Bibliographic Sources

How best should I reference source materials?

- How should I link to a table?
- Can we archive these links at compile time?
- Would we draw the same conclusions?
- Repeatability
- DOI, ISBN, URL

Publishing Research Metadata

So, I have now prepared roughly 100 slides with pen and paper, and now need to lookup which references I browsed, bookmarked, and slept on.

- Tagged Bookmarks (Per Browser)
- Zotero References
  - Archived Resources
  - ISBN, DOI
  - URLs
- Mendeley References
- Course Syllabi
- Course Readings
- Research Journal Databases
- OpenCyc Concept Identifiers
- Wiki Resource Identifiers
- Video URLs
- Media Repositories

That I would like to reference, and draw from

- As footnotes
- As links in blogs & wikis
- As project and career relevant knowledge
- As stable, permalink URLs

Which textual citation serialization?

- Zotero Styles

[TODO:cite]

Tools: Managing Bibliographic Sources

```
Artifacts -> Repository
Metadata -> Citation Knowledge Graph
Groups   -> Citation Knowledge Graph


Bookmarks
   Bookmarklet
   API

Storing Media
   Logical Filesystem Naming
   Interfaces: HTTP GET/PUT/POST, WebDAV, API, JSON, XML
```

RDFa

[TODO:cite]

Gap: Managing Structured Citations

Preprocessing and Serializing Structured Data in ReStructuredText

Data:

- Journals extract and compile bibliography citations in various formats.
- Academia demands citation tracking

Use Cases

- Sort, name, and link to structured citations
- Reference and link to a collection of bibliographic references within *ReStructuredText* documentation.
- Extract structured data from abstract and fulltext *HTML* pages in various databases aggregate and feed journals
- Store research snapshots and research logs
- Store local cache of bibliographical references and citations that can be synchronized, read from, and shared with groups.
- Create a footnote with links to more information
- Create a Zotero Object in a *Zotero* Repository
- Drag and drop a ReStructuredText Citation selection from *Zotero* into a *text editor*, as a bibliography list
- Link to a global citation identifier in *ReStructuredText* inline and within footnotes and citations
- Link to *DOI*, *ISBN*, and *WorldCat*
- Sort and format bibliograpy entries for a given *citation* format
- Archive bibliographical reference repository snapshots in a
  - *Curriculum Vitae*

  - *Paper*
  - *Blog*
  - *Wiki*
  - *Phone*
  - *Tablet*

Standards

- *Microdata*
- *Permalink*
- *RDFa*

Zotero

- http://github.com/zotero
- transactions
- translation server
- pyzotero collection API

TODO:cite

Note: *Zotero* will export a resource collection to *BibTeX*

ReStructuredText Citations

Managing structured citation references

- *ReStructuredText* Syntax
  - Citation
  - Reference
  - Footnote
- *sphinxcontrib-bibtex*
  - Reference *BibTeX* citations in *Sphinx*-flavored *ReStructuredText*
  - TODO: .. pypi: sphinxcontrib-bibtex

Convenient globally non-unique hashed identifiers.

Maintaining Bibliographic citations with BibTeX

Managing citations as structured data with a plain text lightweight markup format.

Interface

```
.. reference a citation from sphinx
.. bib_identifier = {user}/{collection}/{docid}
..                  {group}/{collection}/{docid}

:bibref:`bibliographical reference <bib_identifier>`
```

```
.. format(get_metadata(id) for id in bib_identifiers)

.. bibligraphy::
   :format: <Citation Format>
   :sources:
      [ Reference Collection URLS ]
   :namespaces:
      [ Reference Collection Namespaces ]
```

Note: **:pypi:‘sphinxcontrib-bibtex‘** solves for part of these requirements

Note: Footnote *naming* simplifies reference lookup.

[TODO:cite]

Gap: Sphinx Widont

Jinja2 Typography extensions

Gap: TinCan Server Python Reference Implementation

Objective: Record structured data from event-based *Web Hooks*

- *Python*
- *WSGI*
- Store, verify, and forward *upstream activity metrics*
- Design Q&A Quiz *UI*
- Draw heat map

Python API Tools

- **:pypi:‘Cornice‘**
- **:pypi:‘Tasty‘**

Javascript Activity Metrics

- TODO

TODO:code

Gap: Security: Student API Access Control

How do I share this *API* with a limited set of people?

Access Control

- Reverse Proxy
- *API Token*
- *OAUTH*

Gap: ReStructuredText Outlines and S5 Presentations

Chunking for Presentation and Delivery

Point outlines which map into *slides* of a *document* in *serial order* containing *references* to sources of information regarding the *contextual subject* being discussed.

Slides have a *next* and *previous* and can follow a *question* & *answer* format

Note: presentation timing with sliding windows

Gap: Sphinx Glossary Thesarus

Linking words together.

2.5 License

Date 2012

This report was prepared for English 3980 Technical Writing at the University of Nebraska Omaha, Fall 2012. The views expressed herein do not necessarily reflect the views, positions, or policies of the University of Nebraska. No official endorsement by the University of Nebraska is intended or should be inferred.

This report is a non-profit scholastic work. Third-party content included, cited, and referenced in this report is considered to be fair-use in regards to 17 U.S.C. § 10 and the Copyright Act of 1976.

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License Permission to **reprint**, **share**, and **remix** this report is not necessary. The suggested citation is:

```
Turner, Wesley. *Self-directed Learning with Online Resources*,
   Omaha, NE, USA. 2012.
```

BibTex:

```
@techreport{this,
   author      = "Wesley {Turner}",
   title       = "Self-Directed Learning with Online Resources",
   institution = "WRD",
   year        =  2012,
   address     = "Omaha, NE, USA",
}
```

CHAPTER 3

Process

The `process` of developing this report.

CHAPTER 4

Glossary

4.1 Glossary

Note: The terms listed in this *glossary* are listed in the *genindex*.

4.1.1 Sequence Development

sequencing developing *paths* of *edges* between *node resources*.

Often augmented with *authoring tools*

authoring Creating and synthesizing *sequences* of *resources* like:

- *documents*
- *learning objects*

naming assigning unique identifiers to concepts, objects, and categories

namespacing TODO

```
# Dots:     path.to.resource
# Slashes:  path/to/resource
# Hahes:                    #url_fragment``
```

tagging adding attribute *edges* between *resources* and tag strings, which can be *namespaced URLs*. Tags can denote *categories*

Example in *JSON*

```
{
   "url": "http://example.com/ns/products/XYZ_123",
   "title": "XYZ_123",
   "tags": ["Widgets", "XYZ_Widgets"]
}
```

Often augmented with *annotation tools*

linking Adding *edges* between *nodes* of *resources*

Often augmented with *authoring tools*

software development

Bundling required *resources* and *components*

optimizing finding optima for making decisions that better achieve objectives

publishing sharing *document* and *Linked Data resources* in order to benefit from *collaborative feedback*

interfacing requesting and sharing *resources*

4.1.2 Graphs

Graph A network of *vertices* and *edges*. May have a *name*

Category TODO

Schema A set of *categories* and *attributes*

Examples:

- *XSD*
- *RDF*
- *Markup Languages*

Vertex A node in a *graph*

Edge A connection between *vertices*. Also called a *link*.

Path A sequence of *edges* between *vertices* of a graph

Feedback TODO

4.1.3 Web Standards

Resource TODO. An object with content, a *URL*, and *metadata*

Examples:

- *HTML*
- *Document*
- *Web Video*

WWW World Wide Web. *Graph* of *HTML Document* and *Resource Vertices* with *URL Edges* shared over *HTTP*

Web See: *WWW*

W3C World Wide Web Consortium. The main international standards organization for the *WWW*.

Web Standard TODO. Standard defined by a standards-making body such as *W3C*

SGML Standard Generalized Markup Language

PDF Portable Document Format

URL Uniform Resource Locator

URI Uniform Resource Indicator

HTTP Hypertext Transfer Protocol. Standard *request* /*response* protocol for the *web*.

HTTP Request *HTTP* Request with a type, headers, and a body

Types:

- GET
- POST

- PUT
- DELETE

Example:

```
GET /ns/products/XYZ_123 HTTP/1.1
User-Agent: browsername
Host: example.org
Accept: application/json
```

HTTP Response *HTTP* Response with a response code, headers, and a body

Example Response Codes:

- 200: OK
- 404: Not Found
- 500: Server Error

Example Response:

```
HTTP/1.1 200 OK
Server: servername
Content-Type: application/json
Content-Length: 172
Connection: keep-alive

{"title":"Document Title", "author": ... }
```

TODO:cite

HTML Hyptertext Markup Language.

Derived from *SGML*

Often served over *HTTP*

Example

```
TODO: doctype
<html>
   <head>
      <title>Document Title</title>
      <meta author="Document Author"/>
   </head>
   <body>
      <h1>Document Title</h1>
      <p>... Document Content ...</p>
   </body>
</html>
```

XML Extensible Markup Language. Derived from *SGML* and *HTML*

Example

```
TODO: XMLNS
<object>
   <dc:title>Document Title</dc:title>
   <dc:author>Document Author>/dc:author>
   <content>... Document Content ...</content>
   <year>2012</year>
</object>
```

XHTML *XML*-compliant *HTTP*

Namespace A *URL* for a set of *resources* within a *schema*.

Examples in *Turtle* syntax

```
@prefix rdfs: http://TODO/TODO/TODO
@prefix ex: http://example.org/ns/example/
@prefix products: http://example.com/ns/products/
```

Examples in *XHTML* syntax:

```
TODO
```

JSON *JavaScript* Object Notation.

Example

```
[
 { ’dc:title’:    ’Document Title’,
   ’dc:author’:   ’Document Author’,
   ’content’:     ’... Document Content ...’,
   ’year’:        2012},
 {’dc:title’:’Document N’,’content’:’Hello World’, ’year’:2012}
]
```

Web Hooks *HTTP* Push Notifications

4.1.4 Linked Data Science

Data Science TODO

Metadata Data about data: *attributes* and *edges*

Examples:

- `dc:title` – Dublin Core Title Attribute
- `dc:author` – Dublin Core Author Attribute
- `last_modified`

Key A hashable identifier for a record *value*.

Example:

```
key = http://example.org/ns/products/XYZ_123
```

Value A value stored with a *key*

Example

```
database = {
   ’http://example.org/ns/products/XYZ_123’:    # KEY
      {
      ’type’:’ex:Widget’,                       # VALUE
      ’rdfs:label’:   "Product XYZ_123"
      ’ex:linksWith’: [ ex:XYZ_Widgets ],
      },
}
database.get(’http://example.org/ns/products/XYZ_123’)
database[’http://example.org/ns/products/XYZ_123’]
```

Entity Attribute Value A flexible data storage pattern.

<*entity*> <*attribute*> <*value*>

Triple Data-model of *RDF*

<*subject*> <*predicate*> <*object*>

Subject *URL* Subject of a triple. Also: *Key* and *Entity*

Predicate *URL* predicate of a triple. Also: *Key*

Object Object or *value* of a triple.

Attribute A factual assertion about a *Resource*.

A *predicate* and an *object* about a *subject*

Example with *Triples* in *Turtle* syntax:

```
@prefix rdfs: http://TODO/TODO/TODO
@prefix ex: http://example.org/ns/example/
@prefix products: http://example.com/ns/products/

products:XYZ_123
   a ex:Widget ;
   ex:linksWith ex:XYZ_Widgets ;
   rdfs:label "Product XYZ_123" ;
   .
```

Ontology A structured set of *Attributes* and *edges* between *concepts* in a *named graph*

RDF Resource Description Framework. *W3C* *triples* metadata data-model. Often expressed as *XML*

Turtle Lightweight syntax for expressing *RDF* *triples* (*.ttl*, *.n3*)

TriG Syntax extension for expressing *named graphs* in *turtle*

Microdata TODO. *Markup syntax* for expressing structured data.

FOAF Friend of a Friend *RDF ontology*

DOAP Description of a Project *RDF ontology*

OEMBED Authoring feature for automatically identifying and *linking* to *resource URLs* on sites that support *microdata metadata*

Linked Data Data *resources* linked through the *WWW* using *structured attributes* of various *ontologies*

Linked Open Data *Linked Data* shared as *Data sets* with *Open License* terms

Examples:

- http://dbpedia.org
-

TODO:Cite LODCloud

4.1.5 Education

STEM Science, Technology, Engineering and Mathematics

Curriculum A course or courses of study required for meeting objectives

Theory TODO

Process TODO

Knowledge TODO

Wisdom TODO

4.1.6 Learning

Online Learning Learning delivered over *web channels*

Learning Object "Any entity, digital or non-digital, that may be used for learning, education, or training" –IEEE 1484.12-1-2002

A learning *resource*.

Learning Activity TODO

Learning Assessment Documenting educational progress

LMS Learning Management System. An application for creating and delivering courses and training. "Limbs"

Examples:

- http://blackboard.com
- TODO: http://moodle.org
- TODO: http://sakaiproject.org

LCMS Learning Content Management System. Authoring and publishing workflows to support content for a *Learning Management System*

ADL Advanced Distributed Learning Initiative

SCORM Sharable Content Object Reference Model. Based on *XML*

CLCIMS Computer Learning Content Information Management System: *SCORM*-compliant.

TinCan TinCAN API

"Next Generation *SCORM*"

Web Hooks for *learning activity* metrics

LRS Learning Record Store. A repository for *TinCan learning activity* records.

Can integrate with an *LMS* or *LCMS*

OpenCourseWare TODO

MOOC Massive Open Online Course. Large scale *distance learning* course offered *at scale* through the *WWW*

Examples:

- *Coursera*
- *EdX*

Scalability TODO

4.1.7 Tools

Browser An application for retrieving, presenting and traversing *web resources* like *HTML Documents* over *HTTP*.

Responsible for processing *JavaScript*.

Web Server Software for handling *HTTP* requests over the *web*

Often placed in front of a *Web Application Server*

Web Application Server Software service for hosting web applications that serve *resources* over *HTTP APIs* as content types like `text/html`, `application/json`, `text/xml`. TODO

Interface Standards:

- *WSGI*
- *OSGI*

Service **Business Service**

TODO

Information Systems

A locally or remotely hosted application for solving part of a process.

API

An *API* web service.

API TODO Programming Interface.

An application that responds to a standard set of *requests* and returns a standard set of *responses*

Elements:

- Authentication Keys
- Authorization
- *Error Codes*
- *Resource* Schema
- *Web Service* Definitions

Repository A *version-controlled* folder of file *resources*

Version Control System System for storing changesets to a *Repository* Also *Revision Control System (RCS)*

Examples:

- *Distributed Version Control System*

DVCS Distributed *Version Control System*.

Advantages:

- Branching
- Tagging
- Offline

Examples:

- *Git*
- *Mercurial*

Git *Version Control System*

- TODO http://github.com/mirror/kernel
- TODO http://

Mercurial *Version Control System* written in *Python*

- http://hg.python.org
- http://hg.mozilla.org

Version Control Service Hosted *Version Control System* for storing *Repositories*

Examples:

- http://github.com
- http://bitbucket.org

Scripting Language Third generation programming language.

Examples:

- *JavaScript* (*.js*)
- *Python* (*.py*)
- *Ruby* (*.rb*)
- *Perl* (*.pl*)

JavaScript A *scripting language* which can be interpreted client-side in a *Browser* locally as a *script* or server-side in a *Web Application Server*. (*.js*)

Python A *scripting language* which is compiled and/or interpreted locally as a *script* or server-side in an *Web Application Server*

4.1.8 Research Tools

4.1.9 Authoring Tools

Authoring Tools Examples:

- *Text Editor*
- *Markup Language*

Document TODO. A *resource vertex* in a *resource graph* containing textual content often stored in a structured *markup language*.

Examples:

- *HTML* (*.html*)

Markup Language Textual Markup Language for expressing *documents* with *content* and *presentation*.

Examples:

- *ReStructuredText* (*.rst*)
- *LaTeX* (*.tex*)
- *BibTeX*
- *PDF* (*.pdf*)
- *HTML* (*.html*)
- *XHTML* (*.xhtml*)
- *HTML5*

- *Markdown* (*.md*)
- *MediaWiki Syntax*
- *JSON*
- *XML* (*.xml*)
- *DocBook* (*.xml*)
- *OpenDocument (OpenOffice)* (*.odf*)
- *OpenXML (MS Word)* (*.docx*) # TODO

Text Editor Examples:

- *vim*
- *emacs*
- *gedit*
- *notepad*
- *notepad++*

ReStructuredText A lightweight *Markup Language*. Also: *ReST* and *RST*. (*.rst*)

Example:

```
.. header:: Document Header
.. meta::
   :description lang=en: Document Description
   :author: Document Author

.. contents:: Table of Contents
   :depth: 1

Intro
======
.. note: This is a `note directive <note_directive>`_

.. _note_directive: http://docutils.sf.net/

Background
-----------
.. Document Content ...

Glossary
=========
.. glossary::

   ReStructuredText
      A lightweight :term:`Markup Language`
```

SeeAlso:

- http://docutils.sf.net/docs/user/rst/demo.txt
- http://docutils.sf.net/docs/user/demo.rst

LaTeX Plaintext typesetting *Markup Language*

Example:

```
TODO
```

BibTeX Language and system for managing Bibliographic References in *LaTeX* syntax

```
@techreport{this,
   author      = "Wesley {Turner}",
   title       = "Self-Directed Learning with Online Resources",
   institution = "WRD",
   year        =  2012,
   address     = "Omaha, NE, USA",
}
```

PDF Portable Document Format

rst2pdf *ReStructuredText PDF* publisher.

Output formats:

- *PDF*

Sphinx *RestructuredText* documentation publisher.

Output Formats:

- *HTML*
- *JSON*
- *PDF*
- *LaTeX*

Examples:

- http://docs.python.org
- http://packages.python.org
- http://readthedocs.org
- http://sphinxdoc.org

TODO:cite

4.1.10 Review Tools

4.1.11 UI/UX Design

Interface TODO

UI User Interface

UX User Experience

4.1.12 Cloud

Cloud TODO

Grid TODO

Stack TODO

Distributed Computing TODO

4.1.13 Collaboration Engineering

Collaboration working together to create, share, and improve *resources*

Collaboration Engineering TODO

Six Patterns of Collaboration

1. *Generate*: Fewer to more concepts
2. *Reduce*: Many concepts -> focus
3. *Clarify*: Less -> More Shared Understanding
4. *Organize*:
5. *Evaluate*: Less -> More Value Understanding
6. *Build Consensus*: Less -> More Willingness to Commit

TODO:Cite

Generate Fewer to more concepts.

Six Patterns of Collaboration #1

Reduce Many concepts -> focus

Six Patterns of Collaboration #2

Clarify Less -> More Shared Understanding

Six Patterns of Collaboration #3

Organize TODO

Six Patterns of Collaboration #4

Evaluate Less -> More Value Understanding

Six Patterns of Collaboration #5

Build Consensus Less -> More Willingness to Commit

Six Patterns of Collaboration #6

Seven Layer Model

1. *Goals*
2. *Products*
3. *Activities*
4. *Patterns*
5. *Techniques*
6. *Tools*
7. *Scripts*

Goal TODO

Product TODO

Activity TODO See *Learning Activity*

Pattern TODO

Technique TODO

Tool TODO

Script TODO

Comparison Scheme for Collaborative Technology

- *Core Functionality*
- *Access Controls*
- *Alerts/Interrupts*
- *Content*
- *Actions*
- *Synchronicity*
- *Identifiability*
- *Relationships*
- *Persistence*

Core Functionality TODO

Access Controls TODO

Alerts/Interrupts TODO

Content TODO

Actions TODO

See also: *activities*

Synchronicity TODO

Identifiability TODO

Relationships TODO

Persistence TODO

Creative Process

- *Problem Identification*
- *Information Search*
- *Idea/Solution Generation*
- *Idea/Solution Evaluation and Selection*
- *Implementation Planning*

Problem Identification TODO

Information Search TODO

Idea/Solution Generation TODO

Idea/Solution Evaluation and Selection TODO

Implementation Planning TODO

Goal Attainment Paradigm

- Understand Problem
- Develop alternate solutions

- Evaluate solutions
- Make choices
- Make plans
- Take action
- Review

Six Sigma TODO

DMAIC *Six Sigma* process

- Define
- Measure
- Analyze
- Implement
- Control

Define TODO

Measure TODO

Analyze TODO

Implement TODO

Control TODO

Feature Matrix TODO

```
Feature:
   Label
   Description
   Value

Choice:
   Label
   Description
   {Version}

Choice-Feature:
   Feature
   Choice
   --
   Score
   Reason
   Reference URIs
   Cost
   Cost URIs

Display Algorithm:
   for f in sorted(features):
      print(feature)
      for c in sorted(choices):
            print(choice_features((feature,choice)))
```

Note: Categorically enumerated heat map/contour plot with combinatorially optimized feature islands

Note: max-flow algorithms

TODO:cite

4.1.14 Index

Note: These terms are listed in the *Index*.

CHAPTER 5

Index

The *Index* lists the terms in the *glossary*

CHAPTER 6

Search

- *search*

CHAPTER 7

License

Date 2012

This report was prepared for English 3980 Technical Writing at the University of Nebraska Omaha, Fall 2012. The views expressed herein do not necessarily reflect the views, positions, or policies of the University of Nebraska. No official endorsement by the University of Nebraska is intended or should be inferred.

This report is a non-profit scholastic work. Third-party content included, cited, and referenced in this report is considered to be fair-use in regards to 17 U.S.C. § 10 and the Copyright Act of 1976.

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License Permission to **reprint**, **share**, and **remix** this report is not necessary. The suggested citation is:

```
Turner, Wesley. *Self-directed Learning with Online Resources*,
   Omaha, NE, USA. 2012.
```

BibTex:

```
@techreport{this,
   author      = "Wesley {Turner}",
   title       = "Self-Directed Learning with Online Resources",
   institution = "WRD",
   year        =  2012,
   address     = "Omaha, NE, USA",
}
```

A
Access Controls, 54
Actions, 54
Activity, 53
ADL, 48
Alerts/Interrupts, 54
Analyze, 55
API, 49
Attribute, 47
authoring, 43
Authoring Tools, 50

B
BibTeX, 52
Browser, 48
Build Consensus, 53

C
Category, 44
Clarify, 53
CLCIMS, 48
Cloud, 52
Collaboration, 53
Collaboration Engineering, 53
Comparison Scheme for Collaborative Technology, 54
Content, 54
Control, 55
Core Functionality, 54
Creative Process, 54
Curriculum, 47

D
Data Science, 46
Define, 55
Distributed Computing, 52
DMAIC, 55
DOAP, 47
Document, 50
DVCS, 49

E
Edge, 44
Entity Attribute Value, 47
Evaluate, 53

F
Feature Matrix, 55
Feedback, 44
FOAF, 47

G
Generate, 53
Git, 49
Goal, 53
Goal Attainment Paradigm, 54
Graph, 44
Grid, 52

H
HTML, 45
HTTP, 44
HTTP Request, 44
HTTP Response, 45

I
Idea/Solution Evaluation and Selection, 54
Idea/Solution Generation, 54
Identifiability, 54
Implement, 55
Implementation Planning, 54
Information Search, 54
Interface, 52
interfacing, 44

J
JavaScript, 50
JSON, 46

K
Key, 46

Knowledge, 48

L

LaTeX, 51
LCMS, 48
Learning Activity, 48
Learning Assessment, 48
Learning Object, 48
Linked Data, 47
Linked Open Data, 47
linking, 43
LMS, 48
LRS, 48

M

Markup Language, 50
Measure, 55
Mercurial, 50
Metadata, 46
Microdata, 47
MOOC, 48

N

Namespace, 46
namespacing, 43
naming, 43

O

Object, 47
OEMBED, 47
Online Learning, 48
Ontology, 47
OpenCourseWare, 48
optimizing, 44
Organize, 53

P

Path, 44
Pattern, 53
PDF, 44, 52
Persistence, 54
Predicate, 47
Problem Identification, 54
Process, 48
Product, 53
publishing, 44
Python, 50

R

RDF, 47
Reduce, 53
Relationships, 54
Repository, 49
Resource, 44
ReStructuredText, 51
rst2pdf, 52

S

Scalability, 48
Schema, 44
SCORM, 48
Script, 54
Scripting Language, 50
sequencing, 43
Service, 49
Seven Layer Model, 53
SGML, 44
Six Patterns of Collaboration, 53
Six Sigma, 55
Sphinx, 52
Stack, 52
STEM, 47
Subject, 47
Synchronicity, 54

T

tagging, 43
Technique, 53
Text Editor, 51
Theory, 47
TinCan, 48
Tool, 54
TriG, 47
Triple, 47
Turtle, 47

U

UI, 52
URI, 44
URL, 44
UX, 52

V

Value, 46
Version Control Service, 50
Version Control System, 49
Vertex, 44

W

W3C, 44
Web, 44
Web Application Server, 49
Web Hooks, 46
Web Server, 49
Web Standard, 44
Wisdom, 48
WWW, 44

X

XHTML, **46**
XML, **45**