The Texas Course Redesign Learning Object Repository:
Research and Development for a Production System

Migration Plan for the THECB Learning Object Repository

Prepared for

The Texas Higher Education Coordinating Board

by

The Texas Course Redesign Repository Team

under the supervision of
Dr. William E. Moen
<william.moen@unt.edu>

May 2009

Texas Center for Digital Knowledge
College of Information, Library Science & Technologies
1155 Union Circle 311068
Denton, Texas 76203-5017
Phone: 940-565-2473
Fax: 940-369-7872
Web: http://www.txcdk.unt.edu

Table of Contents

Migration Plan for the THECB Learning Object Repository

1. Introduction

The following terminology conventions are used throughout this document:

- *Location* is a server with an instance of DSpace software installed on a port.
- *Source repository* is the repository that hosts communities and collections that should be migrated into another location.
- *Source communities/collections* are communities/collections from the source repository that have to be migrated to the target repository
- *Target repository* is the repository that is planned to host communities and collections being transferred to it from the source repository. Target repository may also host previously existing communities and collections.
- *Repository migration* is the process of transferring content from the source to target repository and process of transferring interface, user accounts, and policies.

This document describes the suggested procedures for migrating DSpace 1.5.x based repositories between locations.

There are two primary methods of transferring the contents of the DSpace repository from one location to another: using DSpace item export/import tools and using database dump and restore.

There are several preliminary considerations that determine which transfer method should be used:

- Extent of the source (whether complete repository or only some of its communities and collections have to be transferred)
- Degree of integration into the target repository (whether target repository hosts other communities and collections beside those migrating from the source repository).
- Types of items stored in the source collections. For the hyperlinked items with determined entry page primary bitstream is normally set in the source repository. Primary bitstream information is not transferable using current version of DSpace item export/import tools. Resetting primary bitstream for each item in target repository might be not efficient. Migration on the database level may be considered in this case.
- Existence of references between items using URI or handles.

The following sections describe migration procedures in details.

2. Preliminary Considerations

For the scenario when target repository is planned to host exclusively the source communities/collections it is recommended to install a new instance of DSpace in target location. Installation procedure is out of the scope of this document and may be found in DSpace 1.5.2 Manual (DSpace Foundation, 2009).

For any scenario the following decisions have to be made prior to performing migration process:

- Determine if persistent identifiers assigned by handle server in the source repository are going to be used without change or they should be reassigned
- Determine if date elements should be preserved for the items from the source repository

Additionally, if target repository is already hosting other communities/collections the following decisions have to be made:

- Determine the place of the source communities/collections among those in the target repository structure
- Determine whether source collections will use browse and search indexes that are different from indexes in target repository
- Determine whether source collections will use Manakin themes that are different from the themes of the target repository
- Determine users, submitters, and administrators for the source collections
- Determine if source collections need to be updated later and need input form for submitters

3. Migration Using Export/Import Tools

3.1. Communities and Collections

Information about communities and collections is stored in DSpace's backend database. Collections in target repository can be set using Manakin's GUI. All properties of the collections except Name are optional. Note that text from the License property is used to populate each submitted item's Original License property (file license.txt attached to each item). If this property of a collection is left empty default DSpace license text is used.

Communities and collections may also be created using DSpace structure builder tool by running it as follows:

```
<dspace>/bin/dsrun org.dspace.administer.StructBuilder -f <source XML file> -
o <output file> -e <eperson email>
```

The XML filer should be structured as in the following example (all elements are optional as well as they may have empty values):

```
<?xml version="1.0"?>
<import_structure>
    <community>
        <name>History</name>
        <description></description>
        <intro>This Discipline includes…</intro>
        <copyright>Course content is…</copyright>
        <sidebar></sidebar>

        <community>
            <name>American History</name>
            <description></description>
            <intro>This Sub-Discipline…</intro>
            <copyright>Course content is…</copyright>

                <collection>
                    <name>1. Units</name>
                    <description></description>
                    <intro>This collection…</intro>
                    <copyright>All items in this collection…</copyright>
                    <license>All content contained…</license>
                    <provenance>This course was developed…</provenance>
                </collection>
                …
…
</import_structure>
```

Communities will be created from the top level, and a map of communities to handles will be returned in the output file. Structure can be arbitrarily deep and supports all the metadata elements that make up the community and collection metadata.

Important Note: THECB LOR project team tested this tool and found that consecutive use of the tool does not update or overwrite existing communities/collections. It rather creates new set of communities/collections. This happens even when output file with IDs generated on the first run of the tool is used. So, the tool is not useful for updates, only for initial creation of the community/collection structure.

3.2. Metadata Registry

Metadata registry is stored in DSpace's backend database. DSpace default installation includes Dublin Core metadata schema and set of elements. Metadata registry serves the whole repository without designation or separation by collections. It should be determined which elements have to be added to the metadata registry of the target repository using source repository's metadata application profile.

Additional schemas or/and elements may be added to the target repository using GUI.

Second method of adding schemas and elements is by using DSpace command line tool. For that, first the medatada registry of the source repository should be exported by executing the following command:

```
<dspace>/bin/dsrun org.dspace.administer.MetadataExporter -f <xml output
file> -s <schema>
```

Example of the XML output file for GEM schema is presented below:

```
<?xml version="1.0" encoding="UTF-8"?>
<dspace-dc-types>
    <dc-schema>
        <name>gem</name>

<namespace>http://www.thegateway.org/about/documentation/metadataElements</na
mespace>
    </dc-schema>
    <dc-type>
        <schema>gem</schema>
        <element>cataloging</element>
        <qualifier>individualcataloger</qualifier>
        <scope_note>Name of the person creating the metadata record
describing the learning object.</scope_note>
    </dc-type>
    <dc-type>
        <schema>gem</schema>
        <element>cataloging</element>
        <qualifier>individualcatalogerdomain</qualifier>
        <scope_note>Domain experts</scope_note>
    </dc-type>
</dspace-dc-types>
```

Output file can be taken over to the target repository and DSpace metadata importer tool can be used to import schemas and elements into the target repository:

```
<dspace>/bin/dsrun org.dspace.administer.MetadataImporter -f <xml input file>
```

Important note: THECB LOR project team tested this tool and found that if importer tool finds that an element from the source repository registry already exists in the target repository this element will be safely skipped. So this tool can be used safely without deleting existing elements from the source file.

Note: Metadata elements are case sensitive.

3.3. Bitstream Format Registry

The bitstream formats recognized by the system are stored in the bitstream format registry in the backend database. This registry serves the whole repository without designation or separation by collections. It should be determined which formats have to be added to the bitstream registry of target repository.

For example, the following formats might be added: Flash, JavaScript, Flash Movie.

This registry can be edited via GUI. Alternatively to using GUI DSpace's registry loader tool may be used to add additional formats:

```
<dspace>/bin/dsrun org.dspace.administer.RegistryLoader -bitstream <XML input
file>
```

Below is the example of the XML input file. This file cannot be generated by exporting from target repository and should be created with an editor.

```
<?xml version="1.0"?>

<dspace-bitstream-types>

  <bitstream-type>
    <mimetype>application/x-shockwave-flash</mimetype>
    <short_description>Flash</short_description>
    <description>Flash Format File</description>
    <support_level>1</support_level>
    <internal>false</internal>
    <extension>swf</extension>
  </bitstream-type>

  <bitstream-type>
    <mimetype>application</mimetype>
    <short_description>Flash Movie</short_description>
    <description>Flash Movie Authoring File</description>
    <support_level>1</support_level>
    <internal>false</internal>
    <extension>fla</extension>
  </bitstream-type>

</dspace-bitstream-types>
```

Important note: THECB LOR project team tested this tool and found that if at least one element from the source file is already in the target registry none of the elements added. So, only formats that are not already in the target repository should be in input XML file.

3.4. Export Tool

The item exporter and item importer can be used to transfer items between DSpace instances. Additionally, set of default or custom scripts should be used to assist in this process. Scripts are used to cleanup items in simple archive format from the system assigned repository specific elements like dates, URI, etc.

Export/import tool uses simple archive format. There is also METS SIP/DIP Packager but it does not work for non-DC schemas because there are no MODS mapping for those elements.

The item exporter can export a single item or a collection of items, and create a DSpace simple archive for each exported item.

To export a collection's items the following command should be run:

```
<dspace>/bin/export --type=COLLECTION --id=<collID> --dest=<dest_dir> --
number=<seq_num>
```

The keyword COLLECTION means that entire collection will be exported. The ID can either be the database ID or the handle. The exporter will begin numbering the simple archives with the supplied sequence number. To export a single item the keyword ITEM and item ID should be used as an argument.

Each exported item will have an additional file in its directory, named *handle*. This will contain the handle that was assigned to the item, and this file will be read by the importer so that items exported and then imported to another machine will retain the item's original handle if necessary.

Exporter creates subdirectories for each item inside of `dest_dir` directory. Subdirectories naming will start with `seq_num` incrementing by 1. Each collection's item will be stored in a subdirectory.

For example, here is content of subdirectory 100:

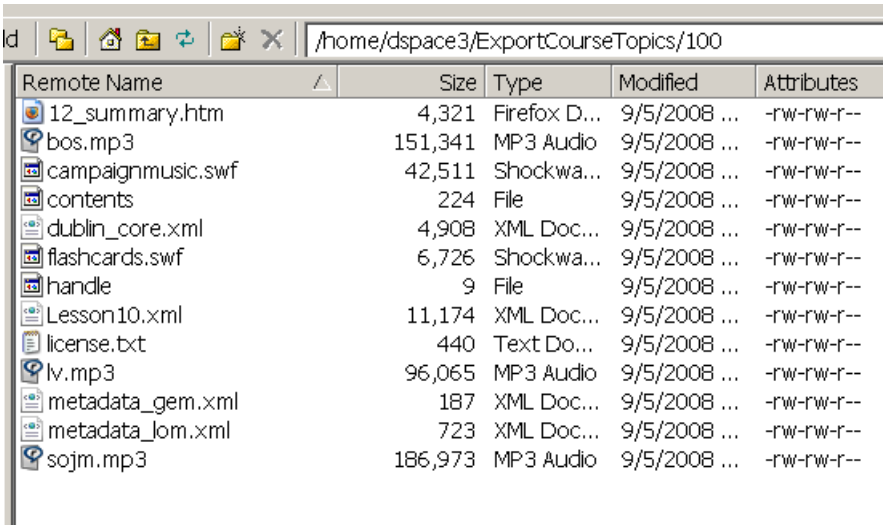

Figure 1: Contents of the subdirectory containing an item is simple archive format

In addition to items content files several system files will be generated. For example, for THECB LOR these files are:

```
handle
contents
metadata_gem.xml
metadata_lom.xml
dublin_core.xml
```

Content of these files is presented and discussed below.

(1) **handle**
Numbers represent suffix for item's URI in the source repository: http://[alias]/handle/2188/242
This file can be removed by the migration script (see following section) if it is not necessary to preserve the same handles values for items in the target repository. If this file is removed new handle will be assigned by the import tool.

```
--------
2188/242
--------
```

(2) **contents**
This file lists all the files in a simple archive format and bundles they belong to

```
-------
sojm.mp3   bundle:ORIGINAL
12_summary.htm bundle:ORIGINAL
campaignmusic.swf bundle:ORIGINAL
flashcards.swf bundle:ORIGINAL
Lesson10.xml   bundle:ORIGINAL
lv.mp3 bundle:ORIGINAL
bos.mp3bundle:ORIGINAL
license.txt    bundle:LICENSE
--------
```

(3) **metadata_gem.xml**
This file contains GEM metadata

```
-------
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<dublin_core schema="gem">
  <dcvalue element="cataloging" qualifier="individualCataloger">Serhiy
Polyakov</dcvalue>
</dublin_core>
--------
```

(4) **metadata_lom.xml**
This file contains LOM metadata

```
---------
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<dublin_core schema="lom">
  <dcvalue element="classification" qualifier="taxon">History</dcvalue>
  <dcvalue element="classification" qualifier="taxon">United States
History</dcvalue>
  <dcvalue element="educational"
qualifier="interactivityType">Mixed</dcvalue>
  <dcvalue element="educational" qualifier="interactivityLevel">Low</dcvalue>
  <dcvalue element="educational" qualifier="difficulty">Low</dcvalue>
  <dcvalue element="educational" qualifier="typicalLearningTime">20
minutes</dcvalue>
  <dcvalue element="technical" qualifier="requirement">Flash</dcvalue>
  <dcvalue element="technical" qualifier="requirement">Windows Media
Play</dcvalue>
</dublin_core>
---------
```

(5) **dublin_core.xml**
This file contains DC metadata. Values of some elements are shortened for the presentation purposes.

```
----------
<?xml version="1.0" encoding="utf-8" standalone="no"?>
```

```
<dublin_core schema="dc">
  <dcvalue element="contributor" qualifier="author">McMichael,
Kelly</dcvalue>
  <dcvalue element="date" qualifier="accessioned">2007-10-
12T00:05:47Z</dcvalue>
  <dcvalue element="date" qualifier="available">2007-10-
12T00:05:47Z</dcvalue>
  <dcvalue element="date" qualifier="issued">2007-10-12T00:05:47Z</dcvalue>
  <dcvalue element="identifier"
qualifier="uri">http://hdl.handle.net/2188/242</dcvalue>
  <dcvalue element="description" qualifier="abstract">This topic summarizes
the information…</dcvalue>
  <dcvalue element="description" qualifier="provenance">Submitted by Svetlana
Barnes (sbarnes@lis.admin.unt.edu) on 2007-10-12T00:05:47Z
No. of bitstreams: 7
sojm.mp3: 186973 bytes, checksum: e0afdefafeac7be02a5fdecb553991ce (MD5)
12_summary.htm: 4321 bytes, checksum: 4128a16f849cca4af24bb330572c734b (MD5)
campaignmusic.swf: 42511 bytes, checksum: 32732d2cc18bd4af415a3f9097ff7d20
(MD5)
flashcards.swf: 6726 bytes, checksum: 44a2c356f4182366b41d24c0ea666bf5 (MD5)
Lesson10.xml: 11174 bytes, checksum: ec66a83c9d46a123520b5e42eb9ecf05 (MD5)
lv.mp3: 96065 bytes, checksum: 4957b4dc1bccbec1e2eaa0a8c5ec90b1 (MD5)
bos.mp3: 151341 bytes, checksum: 0ec97b1d616d67fb1d0ef1bbae257b45
(MD5)</dcvalue>
  <dcvalue element="description" qualifier="provenance">Made available in
DSpace on 2007-10-12T00:05:47Z (GMT). No. of bitstreams: 7
sojm.mp3: 186973 bytes, checksum: e0afdefafeac7be02a5fdecb553991ce (MD5)
12_summary.htm: 4321 bytes, checksum: 4128a16f849cca4af24bb330572c734b (MD5)
campaignmusic.swf: 42511 bytes, checksum: 32732d2cc18bd4af415a3f9097ff7d20
(MD5)
flashcards.swf: 6726 bytes, checksum: 44a2c356f4182366b41d24c0ea666bf5 (MD5)
Lesson10.xml: 11174 bytes, checksum: ec66a83c9d46a123520b5e42eb9ecf05 (MD5)
lv.mp3: 96065 bytes, checksum: 4957b4dc1bccbec1e2eaa0a8c5ec90b1 (MD5)
bos.mp3: 151341 bytes, checksum: 0ec97b1d616d67fb1d0ef1bbae257b45
(MD5)</dcvalue>
  <dcvalue element="format" qualifier="none">Multimedia</dcvalue>
  <dcvalue element="language" qualifier="iso">en_US</dcvalue>
  <dcvalue element="publisher" qualifier="none">Texas Higher Education
Coordinating Board</dcvalue>
  <dcvalue element="relation"
qualifier="isPartOf">http://hdl.handle.net/2188/16</dcvalue>
  <dcvalue element="rights" qualifier="none">This learning object was
developed…</dcvalue>
  <dcvalue element="subject" qualifier="none">Democracy</dcvalue>
  <dcvalue element="subject" qualifier="none">Political Parties</dcvalue>
  <dcvalue element="title" qualifier="none">Lesson 10: Summary</dcvalue>
  <dcvalue element="type" qualifier="none">Lesson</dcvalue>
  <dcvalue element="contributor" qualifier="affiliation">University of North
Texas</dcvalue>
  <dcvalue element="instructionalMethod" qualifier="none">Multimedia
Instruction</dcvalue>
  <dcvalue element="audience" qualifier="educationLevel">Undergraduate Lower
Division</dcvalue>
  <dcvalue element="date" qualifier="published">Oct. 2007</dcvalue>
  <dcvalue element="rights" qualifier="accessRights">This learning object can
be used…</dcvalue>
```

```
  <dcvalue element="rights" qualifier="license">Use and reuse of this
learning object…</dcvalue>
  <dcvalue element="rights" qualifier="rightsHolder">The Texas Higher
Education Coordinating Board. P.O. Box…</dcvalue>
  <dcvalue element="provenance" qualifier="none">Created by the University of
North Texas with funding by…</dcvalue>
</dublin_core>
-------
```

3.5. Script for Preparing Items for Import

Each `dublin_core.xml` file contains metadata that is automatically added by DSpace during submission or import of items. These elements are as follows:

`dc.date.accessioned`
`dc.date.available`
`dc.date.issued`
`dc.identifier.uri`
`dc.description.provenance` (value *Submitted by...*)
`dc.description.provenance` (value *Made available…*)

Import (without pre-processing with a script) with import tool described in the following section adds additional instances of the following elements and duplication is taking place:

`dc.date.accessioned`
`dc.date.available`
`dc.identifier.uri` (if file `handle` is removed from source)
`dc.description.provenance` (value *Made available…*)

Note that new `dc.date.issued` will only be assigned (as current date/time) if not already present in the `dublin_core.xml`, in other words only, one instance of this element will exist.

Note that value of handle for `dc.identifier.uri` will be taken from the file `handle` if this file is present and previous instance of `dc.identifier.uri` will be removed. If file `handle` is not present additional instance of `dc.identifier.uri` will be added with auto generated handle value.

In order to avoid the duplication when item is imported to the target repository the following elements can be removed from `dublin_core.xml` file:
`dc.date.accessioned`
`dc.date.available`
`dc.identifier.uri` (unless old value of handle is needed in addition to the new one)
`dc.description.provenance` (value *Made available…*)

Removal of the file `handle` depends of the decision about preserving old or assigning new items' handles.

To do that, prior to running the item importer this script should be run as follows:

```
<dspace>/bin/dspace_migrate <exported item directory>
```

This script will remove these metadata elements:

- `dc.date.accessioned`
- `dc.date.available`
- `dc.description.provenance` (value *Submitted by...*)
- `dc.description.provenance` (value *Made available…*)

- `dc.identifier.uri` -- if it is not the handle (uses URI other than hdl.*)
- removes all `handle` files.

But it will keep:

- `dc.date.issued` -- if the item has been published or publicly distributed before (`dc.date.issued` <> `dc.date.available`)

Then it will be safe to run the item importer.

The script for preparing items can be customized. For example, for the THECB LOR script was customized to remove `dc.identifier.uri` unconditionally by changing the following part of the code

```
| $SED "/element=\"identifier\" qualifier=\"uri\"\>http:\/\/hdl/d" |
```

to

```
| $SED "/element=\"identifier\" qualifier=\"uri\"/d" |
```

3.6. Import Tool

After determining correspondence between source directory with the collection and target collection ID the following command should be run from the command line when logged in as a user of target repository:

```
<dspace>/bin/import --add --eperson=<email> --collection=<collection ID> --
source=<source directory> --mapfile=<output map file>
```

Note that some old elements are deleted with migration script before import (see previous section). New elements with new values were assigned automatically during the import.

Important Note: Primary bitstreams will need to be set manually in the target repository.

4. Migrating Using Database Dump

4.1. General Strategy

Migration using database dump can be used when whole instance of the source repository need to be migrated over to the empty target instance. It cannot be used to migrate individual collections or parts or the source repository. It also cannot be used to add content to previously existing content in a repository.

This method will transfer structure and content of the repository as well as users and policies.

To migrate between instances the following components need to be backed up and restored:

- A snapshot of the PostgreSQL database
- The assetstore directory
- Configuration files
- Interface design and aspects
- Any customizations of the source code

For more details see DSpace 1.5.2 Manual (DSpace Foundation, 2009) Section 6.1.1.
For migration with upgrade see DSpace 1.5.2 Manual (DSpace Foundation, 2009) Chapter 4.

4.2. Migration Steps

4.2.1. Database

The database schemas in versions 1.5.0 through 1.5.2 of DSpace are identical that allows migration within these versions.

Backup DSpace database from the source repository
`pg_dump <dspace_db_name_source_source> > <output_filename.sql>`

Copy dump file `<output_filename.sql>` over to the target repository. Change owner and group if needed:
`chown -R <dspace_user_target>:<dspace_user_target> <dir>`

Prepare target repository. After installation of the fresh version of DSpace at target repository, the database needs to be dropped and recreated with `-T template0` key (otherwise error happens during uploading dump):

```
dropdb -U <dspace_user_target> <dspace_db_name_target>
createdb -U <dspace_user_target> -E UNICODE -T template0 <dspace_db_name>
```

Before restoring, change references from source database name to target database name inside of `<output_filename.sql>`

To load the database into target repository (as postgres user)
#su - postgres
`psql -d <dspace_db_name_target> -f <output_filename.sql>`

Optional (they are set OK after import):
After restoring from a dump primary key generation sequences need to be reset so that they do not produce already-used primary keys.

For this the following sql command should be run:

`psql -U <dspace_user_target> -f [dspace-source]/dspace/etc/update-sequences.sql`

4.2.2. Assetstore

Copy content of
`/dspace/assetstore`
from the source to target instance and change owner.

4.2.3. Handles

If handle prefix need to be updated it can be done by running the `<dspace>/bin/update-handle-prefix` script. The script takes the current and new prefix as parameters. For example:
`[dspace]/bin/update-handle-prefix 2188 123456789`
will change any handles currently assigned prefix 2188 to prefix 123456789, so for example handle 2188/23 will be updated to 123456789/23 in the database.

Note: This script will change any handles used as metadata values (for cross references between items) to handles of owning items if there is no other text in the metadata values except handles.

For example:

Owning item
http://hdl.handle.net/2188/6789

has elemet:
dc.relation.ispartof = http://hdl.handle.net/2188/1234

this value will be changed to
dc.relation.ispartof = http://hdl.handle.net/2188/6789

However, if value is
dc.relation.ispartof = Lesson 2: http://hdl.handle.net/2188/1234
it will not be changed.

4.2.4. Users and Policies

Users of the source repository will be migrated to the target repository. Users can be deleted/updated if necessary.

4.3. Special Considerations When Managing Handles

Base URL for handles for newly submitted items is set in this file:
`<dspace_source>/dspace-api/src/main/java/org/dspace/handle/HandleManager.java`
original:
```
handlePrefix = "http://hdl.handle.net/";
```
example:
```
handlePrefix = "http://txcdk1.unt.edu/THECBLOR_v3_cont/handle/";
```

The branch need to be compiled after change and new dspace-api-1.5.3-SNAPSHOT.jar copied under tomcat.

Handle prefix generated for newly submitted items is set in this file:
`<dspace>/config/dspace.cfg`
original:
```
handle.prefix = 123456789
```
example
```
handle.prefix = 2188
```

Database stores handles and base URLs in two tables:
Table *metadatavalue*, field *text_value*
and
Table *handle*, field *handle*

If only change of base URI is needed replace values in table *metadatavalue*
For example
http://hdl.handle.net/
to
http://txcdk1.unt.edu/THECBLOR_v3_cont/handle/

Run:
```
UPDATE metadatavalue SET text_value=(replace (text_value,
'http://hdl.handle.net/','http://txcdk1.unt.edu/THECBLOR_v3_cont/handle/'));
```

If change of base URI and prefix is needed replace values in tables *metadatavalue* and *handle*

For example
http://hdl.handle.net/2188/
to
http://txcdk1.unt.edu/THECBLOR_v3_cont/handle/123456789/

Run:
```
UPDATE metadatavalue SET text_value=(replace (text_value,
'http://hdl.handle.net/2188/',
'http://txcdk1.unt.edu/THECBLOR_v3_cont/handle/123456789/'));
```

And to replace in table *handle*

2188/*
to
123456789/*

Run:
```
UPDATE handle SET handle=(replace (handle, '2188/','123456789/'));
```

Themes may be linked to communities/collections via handles in this file:
<dspace>/config/xmlui.xconf

Input forms may be linked to collections in this file:
<dspace>/config/input-forms.xml

Indexes need to be recreated/updates after these operations:
index-init

5. Migrating Other Configurations, Interface Design, and Aspects

Copy configuration files or make customizations in target configuration files:
`<dspace>/config`

For example, browse and search indexes are setup in the file:
`dspace/config/config.cfg`
of the source repository

Configuration of these indexes should be copied to the `config.cfg` of the target repository and following command should be run:
`dspace/bin/index-all`

Interface design can be transferred on the theme level. Complete copy of the theme directory should be copied into the target repository. This includes all *.xsl, *xml, and *.css files.

Copy any custom aspects (compiled classes *.jar) from
`<dspace_source>/*/target`
to
`<tomcat>/webapps/<webapp_name>/WEB-INF/lib`

Copy or make any customizations of the source code if these have been done in the source repository:
`<dspace_source>`

References

The DSpace Foundation (2009). DSpace 1.5.2 Manual. Available at: <http://www.dspace.org/index.php/Architecture/technology/system-docs/index.html >