PROCESS EVALUATION OF GENERAL DATA MIGRATION GUIDELINES — A COMPARATIVE STUDY

DENNIS ENG

LINKÖPINGS UNIVERSITET

LIU-IDA/KOGVET-G--10/009--SE
Information systems form the backbone of many organizations today and are vital for their daily activities. For each day these systems grows bigger and more customized to the point where it is heavily integrated in the current platform. However, eventually the platform grows obsolete and the system itself becomes an obstacle for further development. Then the question arises, how do we upgrade the platform while retaining customizations and data? One answer is data migration which essentially is the process of moving data from one device to another. The problems of data migration becomes evident with extensive and heavily customized systems which effectively lead to the absence of any general guidelines for data migration.

This thesis attempts to take a first step in finding and testing a set of general migration guidelines that might facilitate the creation of future migration projects. This is achieved using a comparative analysis of the general migration guidelines contra the process of migrating data between different editions of the Microsoft SharePoint framework. The analysis attempts to find out if the general guidelines are general enough for this migration process and leave it to future research to further assess their generality. This paper will also investigate the importance of using incremental migration and the ability to perform structural change during migration as well as how these issues is handled by the built in migration tool of SharePoint. In the end the general guidelines proved to be sufficient to express the SharePoint migration process and should therefore be used for further research to assess their worth in other projects. In terms of the second issue, the built-in migration tool proved weak in handling either incremental migration nor structural change which is unfortunate due to the benefits these features bring.
# Table of Contents

1. **Introduction** ...................................................................................................................... 5
   1.1 **Background** .................................................................................................................. 5
   1.2 **Purpose** ......................................................................................................................... 5
   1.3 **Abbreviations** ............................................................................................................... 6

2. **Data Migration** ................................................................................................................... 7
   2.1 **Migration Strategy** ....................................................................................................... 7
       2.1.1 **Planning** .................................................................................................................. 8
       2.1.2 **Migration** ............................................................................................................... 9
       2.1.3 **Validation** .............................................................................................................. 9

3. **Microsoft SharePoint** ....................................................................................................... 10
   3.1 **Microsoft SharePoint Foundation 2010** .................................................................... 11
   3.2 **Microsoft SharePoint Server 2010** ............................................................................. 14

4. **Migration to SharePoint 2010** ......................................................................................... 17
   4.1 **Planning** ....................................................................................................................... 17
   4.2 **Upgrade Approaches** ................................................................................................. 19
       4.2.1 **Inplace Upgrade** .................................................................................................... 19
       4.2.2 **Database-Attach Upgrade** .................................................................................. 20
   4.3 **Review and Validate** .................................................................................................... 21

5. **Method** .............................................................................................................................. 22
   5.1 **Migration Process** ....................................................................................................... 22
       5.1.1 **Migration 1** ......................................................................................................... 23
       5.1.2 **Migration 2** ......................................................................................................... 23

6. **Results** .............................................................................................................................. 24

7. **Analysis** ............................................................................................................................. 26
   7.1 **Planning** ....................................................................................................................... 26
   7.2 **Migration** ..................................................................................................................... 30
   7.3 **Validation** ..................................................................................................................... 32

8. **Discussion** ........................................................................................................................ 33
   8.1 **Method** ......................................................................................................................... 33

9. **Conclusions** ....................................................................................................................... 34

Bibliography .................................................................................................................................. 35
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General Migration Guidelines</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Site Hierarchy</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Database Architecture</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>Managing Sites using the Object Model</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Example of a CMDlet</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>Typical Site Structure</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Managing Solutions</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Upgrade Process Overview</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>Inplace Upgrade</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>Database-Attach Process</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>Attaching Content Databases</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>Results of the Publishing Site Migration</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>Results of the Collaboration Site Migration</td>
<td>25</td>
</tr>
<tr>
<td>14</td>
<td>Planning Phase Comparison Overview</td>
<td>26</td>
</tr>
<tr>
<td>15</td>
<td>Supported Structural Change During Migration with the MSMT</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>Unsupported Structural Change During Migration with the MSMT</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>Creation of a Migration Plan</td>
<td>29</td>
</tr>
<tr>
<td>18</td>
<td>Migration Phase Comparison Overview</td>
<td>30</td>
</tr>
<tr>
<td>19</td>
<td>System Provisioning of the Database-Attach Approach</td>
<td>31</td>
</tr>
<tr>
<td>20</td>
<td>Validation Phase Comparison Overview</td>
<td>32</td>
</tr>
</tbody>
</table>
1 Introduction

This thesis was performed in collaboration with Ipendo Systems AB which is an IT-company primarily focused on SharePoint development. Their interest was to investigate the process of migrating sites between Microsoft SharePoint 2007 and the soon-to-be-released successor Microsoft SharePoint 2010.

1.1 Background

This thesis is conducted on the basis that few general migration strategies exist up to date and that existing approaches are either to high level or have yet to be applied in practice [1]. This paper will present a promising set of general guidelines that could serve as a base for future migration projects regardless of platform or data migrated. As a first step of testing these general guidelines, this thesis will investigate the process of migrating data between different editions of the Microsoft SharePoint framework.

Because SharePoint is built upon the Microsoft .NET framework it is recommended to acquire some basic knowledge of C# (or Visual Basic) and ASP.NET prior to initiating the migration process. SharePoint itself is basically a new way of looking at web development and knowledge of these languages only get you so far, the migration process also requires insight into the architecture of SharePoint. The migration tool used for this thesis is developed by Microsoft and there are two basic migration strategies available with this migration tool, inplace and database-attach. Even though only the database-attach strategy is used for this thesis, both strategies will be explained to show their differences.

1.2 Purpose

This thesis aims to investigate the process of migrating from SharePoint 2007 to SharePoint 2010 and to conduct a comparative analysis of the process using the Microsoft SharePoint migration tool with general migration guidelines to investigate if these guidelines applies for this kind of project.

Further this thesis will look at the importance of one issue raised by the stakeholder, namely if the migration tool supports incremental migration or structural change. In the light of general migration theory this paper will look at why this is an important aspect of any migration project.
1.3 Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIS</td>
<td>Internet Information Services</td>
</tr>
<tr>
<td>MOSS</td>
<td>Microsoft Office SharePoint Server 2007</td>
</tr>
<tr>
<td>MSMT</td>
<td>Microsoft SharePoint Migration Tool</td>
</tr>
<tr>
<td>SPF</td>
<td>Microsoft SharePoint Foundation 2010</td>
</tr>
<tr>
<td>SPS</td>
<td>Microsoft SharePoint Server 2010</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WSS</td>
<td>Windows SharePoint Services 3.0</td>
</tr>
</tbody>
</table>

Due to the likely confusion with conventions in this thesis I would like to clarify this matter. *SharePoint* in a stand-alone sense symbolizes the complete framework including both SPF and SPS (or WSS and MOSS). In contrast, SPF represents the free Microsoft Server add-on and SPS the licensed product. WSS refers to *Windows SharePoint Services 3.0* and any earlier versions will be clearly marked. These types of abbreviations are used to enhance readability. The Microsoft SharePoint Migration Tool abbreviation (MSMT) is self-invented.
2 DATA MIGRATION

Data migration is essentially the process of moving data from one device to another. Information systems form the backbone of most modern organizations today and are essential for daily work. The problem that many organizations eventually must face is that their information system has become obsolete. Obsolete systems are sometimes referred to as legacy systems and often complicate further development which can cause multiple problems [1].

- Legacy systems usually run on obsolete hardware that is slow and expensive to maintain.
- Software maintenance can be expensive, because documentation and understanding of system details is often lacking and tracing faults is costly and time consuming.
- A lack of clean interfaces makes integrating legacy systems with other systems difficult.
- Legacy systems are also difficult, if not impossible, to extend.

When the decision to commence a migration process eventually is taken the issues of transferring legacy system data becomes imminent. Years of customizations to the legacy system are not easily moved and integrated onto another platform, although migrations like these are fairly common. In a survey made by Softek, [2] more than 60 percent of the respondents indicated that they migrate data quarterly and 19 percent reported weekly migrations. Despite this routine process over 75 percent of the respondents had experienced problems during data migrations. Common problems included:

- Extended or unexpected downtime
- Data corruption, missing data or data loss
- Application performance issues
- Technical compatibility issues

2.1 Migration strategy

Because of the complexity of the typical migration process and the divergence of information systems today there is no elaborated methodology for data migration [1]. There are, however, some general guidelines, presented in Figure 1, that describes how to successfully perform a data migration project [3].
2.1.1 Planning

The first aspect of any migration process is to analyze the organizational impact related to the migration. Since the nature of the process almost always result in system downtime, most organizations schedule these projects during off-hours. While this will result in a higher cost due to staff overtime, it may also have a negative impact on staff morale. This and the overall reluctance to take IT systems offline for migration sometimes lead to delays in the development of new technology which ultimately costs money and inhibits advancement [2]. Not only do one have to assess the organization impact, but also consult different expertise involved in daily activities around the legacy system. This may include departments such as IT staff, database administrators and business owners [3].

After business impact analysis one has to collect details about the software and hardware of the legacy system. This prevents any data from being missing or corrupted after the migration. Data discovery can be accomplished either manually or automatically (using a discovery tool) [3]. While a discovery tool is preferred over manual discovery such a tool does not always exist for the entire system. Discovery tools in general reduce discovery time and minimize the chance of error.

When the legacy system data has been identified the destination system is identified and mapped to the existing data. There are two basic layouts for data mapping, one-to-one mapping and re-layout. One-to-one mapping use the same layouts for both legacy and destination system. Although this type of mapping often enables a simpler migration, this is also an opportunity to restructure the layout of the destination system [3].
As pointed out by several sources, the most critical aspect of data migration is the acquiring of a solid migration plan [3] [2]. This step is often quite challenging because all system dependencies might not be known at this point. Nonetheless, every migration project should involve planning and a typical migration plan could include:

- Business and operational requirements
- Data to be migrated
- Available migration tools

Another important aspect of the migration plan is to explore possible migration tools available for the current project. Different tools offer different advantages in terms of system performance, downtime and platform support [2]. In practice, migration planning is often an iterative process and variables may change later in the process but one should not neglect the importance of creating one in the first place.

### 2.1.2 Migration

Provisioning is the task of preparing the new platform for data migration [2]. Both hardware and software requirements must be explored and provisioned in the new system. One-to-one mapping usually result in less work while re-layout may prove a more complex task. Before the actual migration takes place it is recommended to perform an pre-migration test to assure the regularity of the migration plan.

The last phase of migration is to perform the actual cutover to the new system. Three different transition methods have been proposed [1]. The cut-and-run strategy is perhaps the most risky maneuver. This strategy involves switching off the legacy system and initiates the new system in a single step which effectively prevents any roll-back action if the cut-over goes bad. The phased interoperability strategy proposes the cutover process to be performed in small incremental steps. While this strategy can result in migration of more manageable parts and better system downtime planning (e.g. nights) many systems is not easily broken down into separated modules. The last proposal is the parallel operations strategy where the legacy- and the new system operate simultaneously. This strategy enables the new system to be properly tested before it is taken into full service and the legacy system is turned off.

### 2.1.3 Validation

After the migration process is complete it is important to review and validate the results. One should always validate if the expected results are met and what did not work as expected [2].
3 **MICROSOFT SHAREPOINT**

The first thing to know about SharePoint is the dual nature of the platform. Although it has been called different names along the years, *SharePoint Foundation 2010* (SPF) represents the core functionality behind SharePoint and is a free add-on to Microsoft Windows Server 2008. In the former version (SharePoint 2007) the corresponding technology was called *Windows SharePoint Services 3.0* (WSS). The other part of the platform is the licensed product *SharePoint Server* which adds even more features. Figure 2 below shows an overview of the product timeline.

![SharePoint product timeline](image)

SharePoint 2010 is the fourth edition in a line of products which begun with the release of *SharePoint Portal Server 2001*. In the first edition Microsoft introduced the concept of a web portal, known as a *dashboard site*, which effectively let users create a centralized point for sharing and managing information [4]. The dashboard site included features such as:

- Document management and publishing
- Version control of documents
- Information search

With the release of *SharePoint Portal Server 2003* the collaboration features was thoroughly improved with lists, calendars and user profiles. Also the customization abilities of the portal sites were enhanced with the introduction of *areas* where users dynamically could add content such as news lists and calendars [5]. Another customization feature improved was the facilitation of sites and lists. Now users with the right permissions could easily create new sites based on templates, which reduced involvement of the IT department.
In 2007 the platform got an extensive upgrade in layout as well as function. One of the bigger improvements was the introduction of master pages and page layouts enabling administrators to develop templates to facilitate the creation of new sites. Another key upgrade was the Microsoft Office integration feature which let users interact with information stored in SharePoint sites (e.g. documents and spreadsheets) directly without having to manually download the content. The new feature also included support for Microsoft Outlook integration where lists, document libraries and calendars could be imported into Outlook and viewed on the local machine [6].

3.1 Microsoft SharePoint Foundation 2010

SPF is basically a framework dedicated to create dynamic websites used for collaboration purposes, often used by organizations as a tool for creating and customizing intranet and extranet portal solutions. The architecture of SPF is based on the concept of a farm. A farm is a set of one or more server computers working together to provide the SharePoint functionality to users. Farms are typically designed depending on estimated workload where the simplest farm may consist of only one server computer acting as both the front-end Web server and the SQL Server database. Also, SPF is based on the Web server Internet Information Services (IIS) which handles incoming HTTP-requests and redirects users to a specific SPF-site [7].

The SPF web interface makes it easy for users to create sites, share files and communicate through discussion boards. These, so called collaboration sites, are highly customizable and enable users to alter the environment according to their needs. Document libraries is an essential part of collaboration sites and enable users to upload various types of documents for viewing and downloading by anyone with the proper permission. SPF is also connected to Microsoft Office and enables documents from the libraries to be opened locally. This functionality is combined with the ability to check in or out documents from the library. Checked out objects cannot be edited by another user until it is checked in again. In addition to document libraries users can create other lists like task lists, calendars and picture libraries.

Another key aspect of SPF is site permissions and the overall site hierarchy. The first important thing to understand is the structural layering (Figure 3). On the top is web applications which basically is a customized IIS site directly mapped to a unique host header (URL). Under every web application is one or more site collections which contain several sites. This type of structural hierarchy gives the administrator lots of freedom in delegating user permissions. In every site users can easily create lists, document libraries and additional pages to share and publish data.
Behind the scenes of SPF almost all data is stored in databases. Although there are many different databases storing specific information, the most prominent and important databases includes the *configuration database* and *content databases*. The configuration database is automatically provisioned for every instance of SPF and stores vital administrative settings. It is essential for the functionality and each instance of SPF can only host one simultaneously. For every web application there needs to be at least one content database to store its data. But each web application can also consist of several content databases sharing all the data, like Figure 4 below.

![Figure 4. Database architecture](image)

Figure 4 shows the basic layout of an web application with multiple content databases. Every content database consist of one or more site collections which should be distributed evenly. Since the smallest element dividable over several databases is site collections, it is important to plan accordingly when provisioning a new farm since site collections can grow large over time.

Even though SharePoint provides an extensive user interface to interact with elements such as sites and lists there are lots of operations not suitable for this type of use. One may be to
handle multiple items in a list to perform certain changes. This type of operation require a different type of interaction and this may be done via the SharePoint object model. The object model is an extensive framework developed to allow administrators to perform virtually any operation available within SharePoint without having to alter the databases directly. You can of course query the database directly to extract/modify information, however this is not the recommended method of choice [8]. Since SharePoint is built upon the .NET framework the object model utilizes the depth of ASP.NET and logical languages such as Visual C# and Visual Basic [9].

One important aspect of the object model relevant for this thesis is the import/export functionality. To perform structural alteration of SharePoint sites one can use these classes to copy and move elements ranging from complete sites to single items in a list. The object model utilizes the object-oriented nature of Visual C# or Visual Basic to handle all elements of SharePoint. That is every site, list or list item is represented as an object with its unique set of methods and properties. The small example in Figure 5 creates a new site collection object out of the site collection at the specified URL and changes the title of the root site.

```
using (SPSite siteCollection = new SPSite("URL"))
{
    SPWeb site = siteCollection.RootWeb;
    site.Title = "New Title";
    site.Update();
}
```

Figure 5. Managing sites using the object model

Every element can be managed in a similar fashion and although methods and properties vary among different objects the basic principle is the same. This is a very useful way of handling elements and provides high flexibility when dealing with large amounts of data. Now, when moving data from one site or site collection to another two classes are used to perform this operation. The SPExport class takes an object of an element, such as a site or list, and stores the data and metadata locally on the machine. In the next step the local file is accessed using the SPImport class and extracted to its new location. While this process may seem needlessly dual, and it is, there is currently no better solution for moving content between sites. The most time economic solution would be to move the data in a single step without having to store the files in-between. But there is also an advantage in this architecture, when moving content between separate farms for instance. If the client executing the export script is not connected to both farms simultaneously the compressed files needs to be manually transferred, which is facilitated by this layout.

There is however an addition way of working with elements of SharePoint which does not involve programming. SPF offers a wide range of commands using Windows PowerShell and
is using the concept of an *cmdlet*. A cmdlet is combining a command with an object on which the command acts [10] and is used to perform various tasks in the SharePoint environment. Many operations available in the object model is also possible through cmdlets even though the object model generally offers a more elaborated interface working with such tasks.

![Figure 6. Example of a cmdlet](image)

Figure 6 shows how to work with cmdlets. This particular cmdlet is testing a content database for references to elements not installed in the current environment and is typically used in the database-attach upgrade process to check the new farm for inconsistencies.

### 3.2 Microsoft SharePoint Server 2010

*SharePoint Server 2010* (SPS) builds on SPF by adding even more features including publishing, which allows organizations to create Internet-facing sites with full branding abilities. SPS seamlessly integrates with SPF to form consistency in both layout and function. The idea behind SharePoint is to facilitate the creation of websites while keeping the database size to a minimum to improve performance. The publishing feature in SPS enables organizations to enhance functionality of sites and fully customize their appearance. In an effort to improve performance with bigger site collections SharePoint is introducing the concept of *master pages* and *page layouts* (Figure 7). Master pages define the overall layout of sites and assure a consistent look and feel across all sites.
In practice a master page is a site template where you incorporate another page, called a content page, which together form the complete site. A content page is also built upon a template, called a page layout. Page layouts uses placeholders that basically are containers of data retrieved from the content database. This architecture improves performance by allowing multiple sites to be created using a single master page and page layout.

Custom master pages and page layouts are sometimes referred to as customizations. But virtually everything that is user developed can be said to be a customization. SharePoint also allow administrators to create custom web parts that can be added to collaboration as well as publishing sites using the web interface. A web part is a small application which users dynamically can add to sites, without any programming skills whatsoever. Web parts are added to predefined zones to ensure that sites stay structurally organized.

Customization are sometimes packaged as solutions which is bundled files that contain one or more customizations such as master pages, page layouts and web parts. The solution package needs to be installed on the current SharePoint farm (Figure 8) to be available for use and this is done using a cmdlet.
After installation of the solution packages, administrators can choose to deploy each to a specific web application or globally to the entire farm. Only when the solution is deployed that web application can use the contents of the solution.
4 MIGRATION TO SHAREPOINT 2010

Although there are several tools for SharePoint site migration, this paper deals only with the Microsoft migration tool (MSMT). This tool is part of a migration process that requires a certain amount of user involvement and knowledge of the SharePoint framework. While this obviously complicates the process for the user, the complexity of SharePoint makes a simple linear migration process problematic. This section will use the terms migration and upgrade interchangeably since this process is both an upgrade (from old to new system) and a migration (moving from one platform to another). An overview of the SharePoint upgrade approach is presented in Figure 9 [11].

4.1 Planning

A SharePoint migration project will often affect several people in an organization and it is therefore important to inform these people of the upcoming upgrade. Both site users and site owners have to know what to expect of the migration process and what steps need to be taken after the migration process is complete. In smaller deployments, the migration team could consist of only a single person but bigger organizations may require briefing of different people with different roles including [12]:

- Database administrators
- Server security team
- Site collection owners
- Site users
- Stakeholders

Communication also affects the efficiency of the migration process and may reduce downtime. As pointed out before one should always seek to minimize downtime as it may have a negative effect on migration cost as well as staff morale [2]. The next recommended operation in the planning phase is the execution of the pre-upgrade checker. This is a
command-line tool designed to search the server farm for potential issues which might hinder the upgrade process [13]. The pre-upgrade checker looks for, among other things:

- **Upgrade readiness.** Tests each server in the farm to ensure that they meet the requirements for upgrading.
- **Installed elements.** Looks from installed customizations such as user created master pages, page layouts and web parts. These elements will have to be manually migrated and installed in the new environment.
- **Orphaned objects.** Reports on sites, lists or pages whose parent no longer exist and thus cannot be accessed. These objects should be repaired prior to migration since their dysfunctional status will remain even after the upgrade.
- **Valid configuration settings.** Reports on invalid or missing configuration settings in the farm.
- **Database requirements.** Checking whether the databases meet the requirements.

Information from the pre-upgrade checker may be used to determine the appropriate upgrade approach. It may also assist in assessing the duration of the migration process since customized elements generally requires more user involvement. The pre-upgrade checker also tests if the current farm has the necessary system requirements to support an upgrade to SharePoint 2010. The fact that obsolete systems often run on obsolete hardware [1] may complicate an upgrade. But as we will see later, one upgrade approach also enables system upgrade. The following table shows the different system hardware requirements between SharePoint versions [14] [15].

<table>
<thead>
<tr>
<th></th>
<th>Office SharePoint Server 2007</th>
<th>SharePoint Server 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>32-bit, single-core, 2.5 GHz</td>
<td>64-bit, quad-core, 2.5 GHz minimum per core</td>
</tr>
<tr>
<td>RAM</td>
<td>1GB</td>
<td>4GB for developer, 8GB for production</td>
</tr>
<tr>
<td>Disk</td>
<td>3GB</td>
<td>80GB</td>
</tr>
</tbody>
</table>

**Table 1. Hardware requirements comparison**

Due to the substantial difference in the system requirements, it is likely that farms running MOSS will need hardware upgrades to cope with SPS. Apart from hardware requirements SPS also require additional software upgrades such as Microsoft .NET framework 3.5 SP1 and Microsoft SQL Server 2008.

Even though there are just two basic upgrade approaches available for migration the overall process also allows for combinations of both to form custom hybrid approaches [16]. The basic approaches is *in-place* and *database-attach* upgrade and will be explained more fully later. One important aspect when planning the migration approach is whether the current platform can support the new system. Since inplace upgrade does not support hardware upgrades one must either plan to perform such operations prior to migration or consider the database-attach approach.
The last step in the planning phase is the restructuring of content or farms. This step is optional but often presents a great opportunity for cleaning up sites and structure to get a fresh start. Restructuring could also be done after the migration process when the system is upgraded. There are however third-party tools available to restructure data during migration which reduces migration time. But in the context of the MSMT administrators must run scripts to perform structural change which takes additional time to develop.

4.2 Upgrade approaches

The MSMT enables users to choose from different upgrade approaches depending on the current situation and goals of the migration process. **Inplace upgrade** is the process of upgrading an existing system to SharePoint 2010 in a single step [16]. While this approach minimizes user involvement and retains farm settings it is also the most risky due to the absence of roll-back abilities. In addition, **database-attach upgrade** provisions SharePoint 2010 onto a new farm and migrates the data in separate steps. This approach obviously involves more user involvement but is more reliable and involves minimum system downtime during upgrade.

4.2.1 Inplace upgrade

Figure 10 presents an overview of the inplace upgrade approach that effectively minimizes user involvement by upgrading the legacy system directly to the new system. This is a tempting upgrade approach because of the small effort and resources needed but it is not for everyone. Since it is upgrading the current system directly it obviously requires this system to meet the hardware requirements for SharePoint 2010 and if the system was designed for its predecessor this may not always be the case. Also since the live system is being upgraded the system downtime will equal the upgrade time and there is no way of undoing the process.

The process is essentially made up of only three major steps and it starts with installing SharePoint 2010 on the current farm. This means that each server in the farm needs to be provisioned consecutively. This step also includes installation of any language packs and such. Next is an optional but recommended step to test the migration before upgrading the live production environment. This is best executed by replicating the current system either physically or virtually and performing the upgrade locally [17]. The last step is executing the migration tool on the live system which automatically updates the farm. This also needs to be done on all servers in the farm [18].
4.2.2 Database-attach upgrade

The database-attach approach in Figure 11 involves setting up a new server farm with SharePoint 2010 separate from the current farm running SharePoint 2007. There are several benefits of the database-attach method even though it need more user attention. Since the new systems is provisioned onto a separate platform both system can be operational simultaneously. Which in practice enables the new system to be thoroughly migrated and tested before the legacy system is closed down. This radically minimizes downtime of the live system and the impact on its users.

As a consequence of provisioning the new system onto a new farm no configuration settings is retained. The configuration database of the old system contains all the customized settings and the MSMT offers no way of retaining those settings other than reapplying them manually to the new farm.

Site customizations needs to be installed on the new farm in order for the migration to be successful. These customizations are often stored in solutions (which was the case in this project at least). The solution packages can then be manually transferred and installed in the new farm. The customizations will however most likely need some changes due to the architectural change between SharePoint versions.

When the customizations are installed on the new farm each content database must be transferred to the new farm. This is done by making a backup of the content databases in the current system then manually transferring and restoring the backup file on the SQL Server of the new system [19].

When the databases are installed on the SQL Server it is recommended to test the databases before upgrading and attaching them to a specific web application. The command-tool utility may be used to test the databases for references to customizations that are not installed in the current environment. The test reveal elements such as:

- Missing master pages or page layouts.
- Missing web parts
- Missing pictures, flash movies etc.
- Orphaned objects
It is recommended to investigate and correct missing elements or orphaned objects prior to upgrade although this may also be done at a later stage. Missing elements will in effect result in sites being broken and not rendering properly. Orphaned objects will seemingly have no visual effect on the farm at all. In contrast they cannot be accessed and will therefore only occupy space in the database, reducing performance.

When the databases are tested the last step of the migration phase is to attach the databases to a specific web application. This web application will be the new host of the content from the old system. While this may be the time to do some structural changes (like adding an additional content database as in Figure 12), one can also replicate the old system and keep that structure intact.

![Figure 12. Attaching content databases](image)

Attaching the databases also initiates the upgrade process where the content of the databases are automatically converted to fit the new system. This step is sometimes rather time consuming when dealing with extensive databases but the tool do support simultaneous upgrade of multiple databases given the capability of the hardware [19].

### 4.3 Review and validate

The recommended post-upgrade steps of the MSMT includes the following [20]:

- Review upgrade log files
- Test environment

For each attach/upgrade operation a log file is created showing the results of the upgrade. This log files should be scanned for errors and warnings to access the overall success of the upgrade. Also the new environment needs extensive testing to check for inconsistencies and flaws. In particular customizations such as master pages or web parts needs to be reviewed and validated to assure that they are still intact and operational.
In practice, the migration was performed using two separate virtual machines. A virtual machine (VM) is a software implementation of a machine (e.g. computer) which executes programs like a physical machine. In this case each VM simulated different operating systems with different version of SharePoint installed. The first machine had Microsoft Server 2003 and MOSS installed, the other used Microsoft Server 2008 and SPS. The first VM was the platform for which the sites where originally developed. Each VM also hosted an instance of Microsoft SQL Server 2008 to manage the SharePoint databases.

Two existing SharePoint sites were then migrated from the obsolete platform Microsoft SharePoint 2007 to the beta version of Microsoft SharePoint 2010, using the database-attach method, as presented in Table 2. Only the database-attach approach was used because it was the only method doable with the resources at hand, namely two pre-provisioned virtual machines with different versions of SharePoint.

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>From platform</th>
<th>To platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Publishing site</td>
<td>Office SharePoint Server 2007</td>
<td>SharePoint Server 2010 Beta</td>
</tr>
<tr>
<td>2</td>
<td>Collaboration site</td>
<td>Windows SharePoint Services 3.0</td>
<td>SharePoint Server 2010 Beta</td>
</tr>
</tbody>
</table>

Table 2. Overview of migrated sites

The publishing sites contained various amounts of customized code, making the migration process less routine and often require more user input and redesign. Requirements for the collaboration site migration included structural alteration of the site. As will be discussed later, the MSMT offers limited support for structural change during migration and therefore the structural alteration of the collaboration site were performed in steps. First the content database were migrated to the new platform whereat a script was created to restructure the elements of the migrated site.

Next, general migration guidelines mainly from [2] and [3] were studied and a comparative analysis was conducted to assess the differences between the general guidelines and the SharePoint migration process. This analysis is performed and discussed in the next section followed by the concluded results.

5.1 Migration process

The following steps were executed in order to migrate existing data to the new platform. These steps only cover the migration phase since the planning phase was similar to both and the review phase is presented in chapter 6.
5.1.1 Migration 1

1. A new web application were created in the new environment (the virtual machine using SPS) to host the migrated sites.
2. Seven solution packages containing the site customizations were moved to SPS.
3. The solution packages were installed and deployed in the SPS environment using the following two cmdlets.
   - \texttt{stsadm -o addsolution \(-\)filename <filename>}
   - \texttt{stsadm -o deploysolution \(-\)name <name>}
4. The content database were copied and moved from the MOSS environment and installed in the SPS environment.
5. The new web application in SPS were tested against the content database to assure that every required customization were installed in the new environment.
   - \texttt{Test-SPContentDatabase \(-\)name <database> \(-\)WebApplication <url>}
6. Finally the content database were migrated and attached to the web application in the SPS environment using the following cmdlet.
   - \texttt{stsadm -o addcontentdb \(-\)url <url> \(-\)databasename <name>}

5.1.2 Migration 2

1. A new web application were provisioned in the legacy system (MOSS). This web application will host the structurally altered site which are to be migrated.
2. A script was created to structurally alter the original web application and move the data to the one provisioned in step 1. The script basically perform the following operations.
   - Moves data to the second web application
   - Create specified site hierarchy
   - Moves sites, document libraries and lists to specified location
   - Correct corrupted paths to links and images
   - Change site names and paths according to specification
3. A new web application is created in the SPS environment.
4. The content database were copied and moved to SPS. Since collaboration sites in general does not contain any customizations, no solution packages needed to be moved and no database test needed to be performed.
5. The database were then attached to the web application in SPS.
After migration, the publishing site basically suffered from four sources of error.

- The configuration file of each web application (web.config) is not migrated and thus must every customization made to this file be manually transferred to SPS.
- The Microsoft SharePoint assemblies needed to be re-attached to the solution packages in order for the code to compile. This is because the storing location of the assemblies is different between MOSS and SPS.
- The style sheets needed to be modified since the core style sheets of SharePoint was partly changed in SPS (SPS has a new user interface). Also the master page needed to be modified to include the ribbon. The ribbon is a new function in SPS, it is a bar at the top of each page containing controls and buttons used for site customization.
- Since no farm-wide settings are retained in a database-attach upgrade, the site needed to be re-indexed in order for the search function to work (represented as step 7 in Figure 13).

All in all the first migration project followed the guidelines but the actual process also became somewhat iterative since some implications was hard to predict in the planning phase. Figure 13 shows the method steps compared to the migration guidelines.
Step 7 was performed after migration since it is not possible to index the migrated site before it has been migrated and this step thus fall outside of the migration guidelines. The collaboration site did not present any flaws in the user interface, most likely because it utilized only standard appearance and no customizations. It did however lead to some conclusions regarding the migration tool.

- Incremental migration has a very limited support in the MSMT. The smallest migrateable part is content databases which sometimes are pretty large.
- Structural alterations of data may be performed either before or after migration. In either case it is a time-consuming process where (in this case) several hundred lines of code was needed. Although this method does offer lots of structural freedom the script is in nature very limited to the specific purpose for which it was created (and thus not very reusable).

This migration process comparison in Figure 14 shows a somewhat different process from the publishing site. Although, this process also proved the need for site indexing after migration to make use of the search function.

![Diagram of collaboration site migration process](image-url)

**Figure 14. Results of the collaboration site migration**
7 Analysis

First, the evaluation process needs some kind of criteria to assess how well the general migration guidelines coincide with the steps of the SharePoint migration process. Since both consist of certain steps, each step in the SharePoint migration process needs to have a counterpart in the general guidelines. If no counterpart is found the processes can be said to be separated from each other. Further, the processes will not be assessed as different if their chronological order is reversed because the process order is not necessarily carved in stone. Different sources of the SharePoint migration process advocate different migration order and some steps may be reversed without any effect on the overall process. This discussion will follow the general migration guidelines step-by-step to assess the degree of correspondence with the SharePoint migration process. Looking at the processes one common aspect of them both is the three main phases plan, migrate and validate. These three phases are in my opinion critical to any migration project and without any of them the process would seem incomplete.

7.1 Planning

General migration planning

SharePoint migration planning

The planning phase of the general guidelines includes business impact analysis, discovery, mapping and the creation of a migration plan. The SharePoint migration planning differs from this layout in some ways (see Figure 15) even though both strategies advocate communicating with upgrade team and determine the likely business impact during migration. The discovery step of the SharePoint migration is represented by the execution of a pre-upgrade checker tool developed by Microsoft. Remember that the general migration process press the importance of an automatic discovery tool to detect and locate data vital to the success of the migration project [3]. Although discovery could be done manually the pre-upgrade checker is a recommended aid, especially when migrating extensive systems.
The pre-upgrade checker also verifies whether the current system meet the hardware and software requirements for an upgrade. This step in the MSMT is very connected to the next since the specifications of the current systems directly will affect the available upgrade approaches.

The next step is mapping and design of the destination system. Although this step is recognized as an important opportunity to restructure the layout of the system [3] it has very limited support in MSMT. The smallest element supported for migration is content databases. This means that there is virtually no support for structural redesign during migration other that attaching content databases to different web applications. This would in effect result in a different site collections being attached to a different web application than they were in the legacy system. But sometimes administrators need to redesign smaller parts, such as individual sites or lists, which is not supported in the MSMT. This type of structural redesign needs to be performed either before or after migration (or using third-party tools). Figure 16 shows a supported redesign.

It is also important to clarify that in each content database are one or more site collections which cannot be moved between databases using only the MSMT. Thus, the site collections in content database 2 will remain in that database even after migration although it is now connected to another web application.
Figure 17. Unsupported structural change during migration with the MSMT

Figure 17 is an example of structural change that is not supported during migration with the MSMT. Moving smaller elements like lists, pages or document libraries between sites or full sites between site collections need additional work. Although there are several ways of performing structural change the collaboration site redesign performed in this thesis utilized import/export scripts to store and move elements. As we have seen this type of work can be rather time consuming because every object first needs to be exported and then imported. Thus, the operation takes twice the time it would have taken if the content were moved to its destination directly. But the benefit is the ability to relocate the objects manually in between if necessary. Another time consuming aspect of structural change is the developing of these import/export scripts, extensive sites needs extensive scripts. It may not seem worth the time and effort to develop such a script for one-time use only. I recommend a solid planning and to identify what content is to be moved where before any actual code is written. Preferable using charts to map the content to its new destination. But even with precision planning one must always be prepared for some minor problems that need ad-hoc solutions, such as paths being broken.

The general migration guidelines advocates creating a migration plan at the end of the planning phase. The MSMT does not explicitly recommend such a step but the process have gathered the required information to create an migration plan. Including the business requirements (in communication with the upgrade team), data to be migrated (using the pre-upgrade checker) and available migration tools (when choosing upgrade approach) as displayed in Figure 18.
Maybe this is something the SharePoint migration process could learn from the general migration guidelines. Even if the information gathered in the migration plan step of the general guidelines is indeed gathered in the planning steps of the MSMT process, the making of a migration plan could possibly help to reflect upon the coming migration phase. So even though the migration planning step is not listed in the MSMT process, it does not invalidate the general guidelines or hinder them from applying for this type of project. The migration plan step should thus be viewed as an optional but recommended operation.

To summarize, the general planning guidelines shows a lot of similarities with that of SharePoint. In fact every step is accounted for except for the creation of a migration plan. But as shown, the components of a migration plan is gathered in the planning process and the migration plan itself should rather be a recommended step. With this in mind my conclusion is that, for the planning phase at least, the general guidelines are valid and sufficiently generic to model a migration planning process within the SharePoint framework.
7.2 Migration

**Inplace upgrade**

- System provisioning
- Optional: Trial upgrade
- Run migration tool

**Database-attach upgrade**

- System Provisioning
- Configure farm settings
- Install customizations
- Install databases
- Optional: Test databases
- Attach databases

---

Figure 19. Migration phase comparison overview

Figure 19 displays an overview of the migration phase of the general guidelines as compared to the different SharePoint upgrade approaches. Even though all of the general steps are accounted for in both approaches they perform migration in quite different ways. The general guidelines start with provisioning the new system and mentions two basic strategies for this. They are a consequence of the strategy picked in the planning phase, one-to-one mapping or re-layout. Both can be applied to a SharePoint migration process depending on the chosen upgrade approach. Inplace upgrade is strictly a one-to-one method since it does not allow structural change at all. Database-attach on the other hand is a more diverse method which support both one-to-one mapping and re-layout. Although as we have seen earlier the database-attach approach does not offer much depth in the ability to make structural changes but can still be used to some degree. Since both SharePoint upgrade approaches demand a system provision step to install required hardware and software, it seems to me as a critical step even though they need different amounts of provisioning.
The database-attach approach need additional provisioning to render successfully. In addition to system provisioning, the process also includes reconfiguring farm settings and installing customizations along with databases. These four steps together comprise the provisioning step of the general migration guidelines (Figure 20). Consequently, the provisioning step is necessary for both SharePoint upgrade approaches although they vary in size with the database-attach being a four-step process.

The pre-upgrade test step is handled quite different between the two upgrade approaches. While the inplace upgrade advocate the quite time consuming action of replicating the complete farm and simulate an upgrade, the database testing step of the database-attach approach is made up of an automatic cmdlet that only takes minutes to execute. Nonetheless they both coincides neatly with the general pre-upgrade step.

The last step of the migration phase is the cutover and it represents the actual data transfer. The cutover operation in each SharePoint approach is significantly different from the other. Whereas the database-attach cutover only attaches and upgrades each content database, the inplace approach upgrades the entire farm with content as well as settings. With the three transition methods described by the general guidelines [1], the cut-and-run strategy shows striking resemblance to the inplace upgrade and is described as an high risk strategy because of the lack of roll-back action. However as we have seen it is a rather automated process which at least should minimize errors caused by the human factor. The database-attach approach may be seen as a combination of the remaining two strategies, phased interoperability and parallel operation. Phased interoperability because of the ability to migrate incrementally, at least to some degree, and parallel operations because both the legacy- and new system operate separately after migration. The nature of the cutover operation does however make it a vital part of the migration process and also serves as a natural ending of the migration phase.

In summary, the migration phase of the general guidelines do encompass all the SharePoint steps, even though each upgrade approach perform differently in terms of execution and the
number of steps required. The provisioning and cutover steps seem equally important and vital to the migration phase while the pre-upgrade test is duly recommended and often greatly facilitate the migration process, but remains optional.

7.3 Validation

The validation phase does not really offer much variation as can be seen in Figure 21. Testing the environment is as critical as the planning and a lot could be learned for validating the results. My experience from this migration project is that the validation phase could in some cases be the most time consuming phase of the migration process, but this is very dependent on the migrated data. Some sites have more customizations than others and generally that leads to a longer validation process which obviously involves lots of troubleshooting and testing.
8 Discussion

8.1 Method

Because the methodology of this thesis might seem overly time-consuming it seems appropriate to discuss other possible approaches and why this method was eventually chosen. Performing a migration project like this requires a certain amount of programming skills as well as knowledge of the SharePoint architecture. While this obviously takes some time to acquire it is also a key aspect for a proper evaluation.

Other suggested methods of evaluation include interviews, surveys and observations. Even if this type of evaluation methods may provide an broader view of the migration process it is not likely it would be sufficient to fully understand the migration process. Qualitative methods may be sufficient for one to understand why organizations choose to migrate legacy system data. But since this thesis aims to investigate one particular migration tool and comparing that to more general migration guidelines, it is essential to understand the underlying functionality of SharePoint itself. For example, one would probably have had trouble understanding the similarities between the parallel operations migration strategy and the database-attach upgrade method only by reading the documentation or through qualitative research. Thus, the technical background is necessary and can, in my opinion, only be acquired by performing an actual migration project.

The optimal solution would likely have been to do both. With the knowledge learned from performing the actual migration project founding a base for qualitative research one would have the tools both to understand how a migration is performed and why organizations need to perform them. But for the context of this thesis, doing both was not possible and therefore the how-to perspective of the migration process was chosen and compared to general migration guidelines. However, the qualitative aspect of migration could indeed lead to some interesting results and should therefore be a subject for future research.
This thesis was conducted to find out two things, (1) if the general guidelines was general enough to cover a SharePoint migration process and (2) to investigate the concepts of incremental migration and structural change to assess why these are important elements of migration and to what degree these are possible using the MSMT.

The first issue has been resolved using a comparative analysis where the outcome points to the fact that the general guidelines are indeed general enough to be used as a template for future SharePoint migration projects. Although certain steps does not coincide completely the processes are equal to a degree that make the general guidelines sufficient for this very purpose. But this conclusion in itself obviously does not make the general guidelines that general, rather it serves as a building block for further research and contributes to the recognition of the guidelines. Future research of the guidelines is necessary to assess their usefulness also in other migration projects.

As for the second issue, we have seen the importance of these elements and also the shortcomings of the MSMT regarding this issue. Incremental migration may lead to benefits such as migration of more manageable parts and to reduced business impact since cutover operations may be split up and executed during off-hours such as nights. The MSMT does not however offer a lot of support for incremental migration. Remember that the smallest object available for migration is content databases and that these databases can grow pretty big with time. Proper architectural planning of the databases could lead to improved cutover performance but other than that the only real solution is the use of third-party tools.

Migration has also proven to be a great opportunity for structural changes. Unfortunately the MSMT has showed to be very unsupportive of such operations as well. The ability to relocate smaller elements internally, such as sites or lists, is not built into the MSMT and such operations therefore need additional work. Thus, structural change could either be done prior to or after migration, or by using third-party tools. Consequently, the MSMT may not be the way to go for organizations looking for a smooth and swift migration process. But it may offset the fact that the MSMT is an integrated part of the SharePoint framework and thus free of charge compared to most third-party tools.
BIBLIOGRAPHY


