Chapter 1

Getting Started

To start with, we’ll introduce ENVY/Developer, go over enough of the basic ENVY/Developer architecture to get you started, and describe in some detail how to set up the environment and configure the options available. All readers will want to review the architecture section. The setup sections will be of particular interest to new users and to administrators responsible for the maintenance of an ENVY installation.

Overview

First, let’s identify what ENVY/Developer is, where it comes from, and what it’s good for. This information will help you understand the architecture and where ENVY’s concepts come from.

What Is ENVY/Developer?

ENVY/Developer is a software engineering environment for Smalltalk programming. Specifically, it provides facilities for team programming and delivering significant-sized applications, built on top of the regular Smalltalk development environment. Smalltalk is widely recognized as being very productive for individual developers, but the standard Smalltalk environment was not designed with the idea of collaborative development in mind. ENVY/Developer extends the basic Smalltalk environment to include facilities for team programming and configuration management.

Technically, ENVY is a generic term that refers to a family of products and technologies. These include ENVY/Smalltalk, ENVY/Developer, ENVY/Replicator, and others. However, ENVY/Developer is the most widely used of these products, and it is common usage to use just the term “ENVY” to refer to ENVY/Developer, or even more specifically to ENVY/Manager. Although this is technically incorrect, it is the common usage and is the standard we follow for this book.

— 1 —
2 Mastering ENVY/Developer

At various points, ENVY implementations existed for all three major commercial Smalltalk platforms. We describe these as follows:

- **VisualWorks**: This includes ObjectWorks/Smalltalk and its successor VisualWorks. These are the direct descendants of the original Xerox Smalltalk and were and are currently owned by Cincom Systems.

- **IBM Smalltalk**: This includes OTI's ENVY/Smalltalk, IBM's IBM Smalltalk, and IBM's VisualAge/Smalltalk products. These products are currently made by IBM and their OTI subsidiary.

- **Digitalk**: This includes Smalltalk/V DOS, Smalltalk/V Windows, and Visual Smalltalk. These products are also owned by Cincom Systems, but have not had ENVY support for some time. We will not consider Visual Smalltalk when talking about ENVY implementations.

The full suite of ENVY products comprises the following modules:

- **ENVY/Developer** is a toolset that exists in a Smalltalk image, implemented in Smalltalk code. For the IBM Smalltalk platform, it consists of four separate products. Currently, only ENVY/Manager is supported for the ParcPlace platform.

- **ENVY/Manager** provides configuration management and version control facilities. This product is the focus of this book. As a product, it is transparent to the user because it is tightly integrated into the development environment. It is not normally possible to load or unload ENVY/Manager.

- **ENVY/Packager** is an image packaging or "striping" tool used to create a deliverable application out of a development image. This is used within products in the IBM Smalltalk family, and also existed historically for Digitalk Smalltalk.

- **ENVY/Swapper** is an object dumper and loader. It can be used to serialize a network of object instances to a file or an arbitrary stream. This is nominally a separate component, but one that must be loaded because ENVY/Manager makes use of it. On VisualWorks platforms, the equivalent BOSS facility is used instead.

- **ENVY/Stats** is an execution profiler that analyzes the time and/or memory used in executing a piece of Smalltalk code.

**History**

ENVY/Developer originated in the late 1980s as a research project at Carleton University in Ottawa, Canada. The project, known as Orwell, was led by Dave Thomas, founder of OTI, and a professor at the School of Computer Science. The concepts introduced in Orwell have served as the foundation of ENVY/Developer for over a decade.

ENVY/Developer was first implemented on Digitalk's Smalltalk/V for MS-DOS, and subsequently evolved to support other members of the Digitalk product family.
It was ported to ParcPlace's VisualWorks, and was used as an integral piece of ENVY/Smalltalk, which then served as the basis for IBM Smalltalk and VisualAge. Ultimately, OTI was acquired by IBM, becoming a wholly owned subsidiary of IBM Canada.

**Architecture**

This section describes the fundamental architecture of ENVY/Developer, and how it differs from both traditional Smalltalk environments and traditional file-based source code control systems.

**The Repository**

A classical Smalltalk environment is intended for an individual developer. The system consists of a collection of files:

- **One or more image files**: These files hold the complete state of a development environment, including all of the compiled code.

- **A sources file**: This contains the source code to the base system. In normal usage, there is only one sources file, and it does not change until there is a new version of the base Smalltalk system.

- **One or more change logs**: For each image file, there is a corresponding change log. Every time code is compiled in the image, the corresponding source code is saved into the appropriate change log. A pointer into the change log is saved with the compiled code, so that its source can be retrieved.

*Figure 1.1: The classic Smalltalk development model.*
In this model, developers work individually. To collaborate with other developers, they would “file out” the relevant code in source code format and send it to them. This allows for collaboration, but does not maintain version information, requires a manual process, and allows information to be lost.

ENVY/Manager augments this model by providing configuration management and version control facilities. All code is stored in a central database rather than in files associated with a particular image. Developers are continuously connected to this database; therefore, changes are immediately visible to all developers. Because all changes are centralized, it is much less likely for code to be lost.

This centralization is very powerful, but it requires significant changes to the basic environment. ENVY cannot be delivered as a traditional standalone component, or as a set of file-ins. Rather, OTI provides a new base image in which fundamental changes have been made to the underlying system. These changes replace the usual change log and sources with an interface to the ENVY/Manager library where all code is maintained. It also structures all source code into one of several types of

![Diagram of ENVY development model]

**Figure 1.2**: The ENVY development model.
components, and provides a complete set of browsers to develop, maintain, and configure these software components. These replace the normal development browsers.

This has important implications for the work process. In the classical model, development was image-centric. With ENVY/Manager the image becomes a discardable by-product of development, secondary to the repository that stores and coordinates all work. An image serves as a private workspace where a user can develop, test, and execute code. The library, to which all users are connected, serves as a shared workspace where users can work separately or cooperatively on the same projects and components.

**Concurrent Development**

The ENVY model of development is also distinct from traditional file-based version control systems, which typically rely on a check-in/check-out development model. With file-based tools the team programming environment looks like some form of shared directory structure. Users check files out into a local directory, then check them back into the system. In Smalltalk, the primary representation of a program is as objects, with text files used only as a backup and interchange form. The ENVY repository acts as an object-oriented (OO) database to hold and re-create these objects. The image serves the purpose of the directory structure, holding together the current state of the various components.

This fine granularity of components enables highly concurrent development because changes can be tracked down to the individual method level, with no necessary correspondence between any of the software components and a disk file. ENVY enables developers to manage components in the repository at any level, ranging from individual methods to entire subsystems, further increasing concurrency. Rather than managing conflicts using a file-based differencing mechanism, it uses component ownership as a fundamental mechanism. Any number of developers can work concurrently on a single component, but one developer is designated as owner of that component and the owner must approve any changes. For more information on this model, see Chapter 3.

In addition to being a multi-user product, ENVY is a multi-platform product. Any ENVY image may be connected to a library, and browse and possibly load components created using other ENVY images running on different platforms or different Smalltalk base images or virtual machines. In fact, sites with multiple users will typically be running images on multiple platforms and sometimes multiple Smalltalk releases, all connected to the same library.

**ENVY Concepts**

This has been a quick overview of the ENVY architecture. For more information about ENVY concepts and architecture in a more formal setting, see Chapter 5. The following sections start to apply these concepts in a more detailed and tutorial style. These
are appropriate for new users, or any users who would like the concepts and development process further explained, with examples and detailed advice.

Installation and Setup

This section talks about installation and setup procedures for ENVY in some detail, including a number of different configuration options. This section isn't intended to replace the setup instructions that come with the product, but rather to supplement them with explanations and advice on the various options available. You should consult the installation instructions and release notes that came with your ENVY installation and your Smalltalk programming environment. We cover both server and client setup. If you are working on an existing project, it’s likely there is a preexisting server setup, and you will simply need to consult with your system administrator on what the appropriate client options are.

Note that some variation exists between different versions of ENVY, and here we primarily discuss the current versions, specifically VisualAge 5.0 and VisualWorks 3.0 and 5i. We discuss both versions of VisualWorks because both are in common use right now and some differences exist between them. Almost all of what we say for VisualAge 5.0 also applies to Versions 4.5 and 4.0.

As a new user, you’ll want to read the sections here to get you up and running as quickly as possible, then move on to Chapter 2, which covers some basic concepts. As an ENVY administrator, you may want to read this section to examine the possible choices you face in configuring the setup for your users. All readers may want to go over this section briefly for things they may not have known about ENVY configuration.

Client Backup

Client Backup

It's always important to have good backup procedures. When installing ENVY, one of the first things you should do is ensure that you have access to a copy of the Smalltalk image as it was distributed (a clean image). One way to do this is to keep a compressed copy somewhere easily accessible. If something goes wrong with your ENVY environment, a clean image is the best place to recover from.

The Repository

The Repository

As described previously under “Architecture,” ENVY uses a central database to store source code. This is normally called the repository but may also be called the ENVY database, the manager file, or manager.dat. A number of these terms come from the repository’s implementation as a single large file, which may be named either manager.dat or with a name encoding the version number, for example, mgr50.dat. This file acts as a database to store the source code and data associated with the various software components.
Accessing the Repository

In the single-user or standalone version of ENVY, the repository file is stored on a local disk and accessed directly. In a team environment, the repository is shared, either through a network file system or the EMSRV server process (the preferred mechanism).

The mechanism used is determined at startup by information specified in the configuration file.

The Configuration File

The configuration file is a simple text file that specifies many important startup parameters for ENVY. The configuration file may be set up automatically as part of your installation, or you may need to edit it manually. In either case, it can be important to look at this file if you need to change or debug your setup.

This file has two primary formats depending on which version of Smalltalk you’re using. In VisualWorks, the configuration file is named ENVY.CNF and contains Smalltalk source code, which is read in, compiled, and executed at startup. Note that the exact contents of this file are different between VisualWorks 3.0 and VisualWorks 5i. In VisualAge, the configuration file is in Windows .INI format, with the same name as the development environment executable (for example, ABF.INI).

At startup, the system reads this file and uses it to set various parameters, including directories to search for various resources. For our purposes, the most important settings are the location of the repository file, whether it is accessed locally or over the network, and optionally the name or IP address of the machine running the ENVY server.

Access Methods

ENVY can access the repository in two different ways: through direct file access (possibly networked) or through a server process called EMSRV.

In direct file access, the Smalltalk image directly accesses the repository file and manipulates it, including obtaining locks. This is most commonly used in a single-user installation, where the file is stored on a local hard drive. In older versions of ENVY, it was also possible to access the repository using network file and locking
services, although not all networks were supported, and UNIX NFS was specifically not supported because it did not support the required locking mechanisms.

EMSRV is an acronym for the ENVY/Manager Server. When using EMSRV access, ENVY does not access the repository file directly, but goes through a server process running on the machine that holds the repository file. This is the preferred mechanism in current versions of ENVY (VisualAge 4.5 and later, VisualWorks 5i and later). Consequently, EMSRV must be used whenever an ENVY repository is to be shared by more than one user or when the repository file is not stored locally on the machine running ENVY. There are several reasons for this change. Not all networks provide reliable locking services, and many subtle bugs can arise with different combinations of networking software and operating systems. Furthermore, the server process can also be more efficient because it can combine multiple file operations into a single request. The biggest disadvantage of this mechanism is that it makes installation significantly more complex.

File Access

File access is extremely simple to set up. You simply need to specify the location of the file. The exact technique for doing this varies depending on your ENVY dialect and version.

Using file access in VisualWorks 5i, the relevant part of your configuration file would look something like this:

```
ENVY.EmLibrary
  defaultName: 'C:\apps\vw5i\envy\manager\manager.dat';
  serverAddress: nil.
```

This specifies the local file path to the ENVY repository, and that there is no server in use. Using file access in VisualWorks 3.0 adds some additional options, so the relevant part of your configuration file would look like this:

```
EmLibrary
  defaultName: 'C:\APPS\VW30\SERVER\MANAGER.DAT';
  serverAddress: nil;
  releaseLockMode: true;
  singleUserMode: true.
```

We have again specified the file name and a nil server, but we have also specified additional options that have become obsolete in VisualWorks 5i. The first of these is releaseLockMode, which was essentially obsolete even in earlier versions but was used to indicate to ENVY that the file system could be relied upon to release locks when the library was closed. Otherwise, the ENVY system would have to manually release any locks. You should leave this setting as true unless the ENVY documentation indicates otherwise for your platform. The other option is to specify whether we’re a single user, and don’t need to worry about contention in the database. If we are, then ENVY can run a little bit more efficiently. Note that if you’ve set this option to true, and someone else attempts to access the database, the attempt will fail.
In VisualWorks 3.0 and earlier, the configuration file also contains the names of shared libraries to be used for network file access and password checking. These will vary by platform, and you should consult the ENVY release notes for which libraries to use.

Using file access in VisualAge, the relevant portion of your configuration file would look like this:

```
[EmLibraryInterface]
DefaultName=C:\apps\wa50\mgr50.dat
ServerAddress=
OpenReadOnly=false
```

This specifies the local file path to the ENVY repository, and that no server is in use. Further, we can specify that we wish to open the repository in read-only mode. This might be appropriate for browsing a repository to which we do not have write access, either because we have insufficient privileges, or because it's on a network partition.

---

**Tip**

Even in ENVY versions that supported repository access over a network file system, you are better off using EMSRV. It's more stable and reliable.

---

**EMSRV Access**

Using EMSRV in VisualAge, the relevant part of your configuration file might look like this:

```
DefaultName=F:\VISUALAGE\MGR50.DAT
ServerAddress=JUPITER
```

This specifies the server file path (see below) and the name of the server, which is a computer on the LAN with the name Jupiter.

Using EMSRV access in VisualWorks 5i, the relevant part of your configuration file would look something like this:

```
ENVY.EmLibrary
defaultName: 'F:\VISUALWORKS\MANAGER.DAT';
serverAddress: '172.30.1.25';
```

This specifies the server's file path to the ENVY repository, and the IP address of the server. Using EMSRV access in VisualWorks 3.0, some additional options exist, and the relevant part of your configuration file might look like the following:

```
EmLibrary
defaultName: 'F:\VISUALWORKS\MANAGER.DAT';
serverAddress: 'JUPITER';
releaseLockMode: true;
singleUserMode: false.
```
Here we have specified the file name and a named server, but we've also specified additional options, which have become obsolete in VisualWorks 5.1. The first of these is releaseLockMode, which was essentially obsolete even in earlier versions, but was used to indicate to ENVY that the file system could be relied upon to release locks when the library was closed. If not, the ENVY system would have to manually release any locks. You should leave this setting as true unless the ENVY documentation indicates otherwise for your platform. The other option is to specify whether we are a single user, and don't need to worry about contention in the database. In this case, because we are setting up with EMSRV, it's unlikely that we intend to use this as a single-user database.

In this example, it's important that the path we provided to the repository is a server path, in terms of the server's drives. So in this example we expect the server to have a physical drive F: with an F:\VISUALAGE or F:\VISUALWORKS directory on it. We don't care if the client has a drive F:, whether physical or networked. Although the configuration file is on the client machine, the repository file path isn't used on the client but is sent directly to the EMSRV process, which runs on the server. The repository path is then used by EMSRV, which interprets it in the context of the server machine it is running on. If, instead of an absolute path name, we used a relative path, such as ..\manager.dat, then this would be interpreted relative to the home directory of the EMSRV process on the server. Note that it is not even necessary for the client to have network access to the repository file when using EMSRV.

The address of the server can be specified either by name or by IP address. In the preceding examples, we have either used the local network name Jupiter, or the IP address of the corresponding machine. We recommend using the IP address. This is less convenient if your networking setup changes, but we have found it more robust in the presence of a variety of network issues.

Also note that when using EMSRV, the repository file must exist on a hard drive that is physically attached to the computer running the EMSRV process. That is, it is not possible to use EMSRV when the directory containing the repository file is network mounted. You should be sure that EMSRV is run on the machine that physically contains the hard drive with the repository file.

**Troubleshooting Installation**

If you can install your basic Smalltalk environment, relatively few things can go wrong with the basic ENVY install. The most common cause of problems is with the EMSRV installation, and that's what we mostly address in this section.

This is not to imply that EMSRV is unstable. As mentioned previously, EMSRV is both more stable and better optimized than file access, and is generally to be preferred. The biggest disadvantage of EMSRV is that there are more things that can go wrong. This section discusses some of the problems that can arise, and provides approaches to debugging them.