# Models to Code Source Files

---

## Overview of Source Code Files

**Legal Notices and Information**

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| 1.0 | April 15, 2017 | Initial release. | GAM |

# Contents

# Introduction

This document is an overview of the source code files provided as part of the book, *Models to Code*. *Models to Code* contains several case study examples. In the interest of brevity, only portions of the examples are contained in the book itself. The completely worked out example code is given here.

## Literate Programs

The example code is presented as a literate program. This means that the source file contains both the descriptive documentation for the example as well as the `pycca` source code which implements the domain. The example source files are also valid asciidoc documents. A readable document, in many different formats (*e.g.* PDF), can be generated from the source using `asciidoc`. The `pycca` source code that implements the model can be extracted from the literate program source file using a tool named, atangle, which is included in this distribution. This process is discussed below.

For all of the examples, this source distribution provides both a PDF of the literate program document as well as all the extracted and generated source files. If you only wish to examine the example documents and code, it is not necessary to install any software.

# Tools

All the tools included in here executables for the three major platforms are provided.

- Linux
- MacOSX
- Windows

```
tools
|-- linux
|   |-- atangle
|   |-- pycca
|   `-- tack
|-- macosx
|   |-- atangle
|   |-- pycca
|   `-- tack
`-- windows
    |-- atangle.exe
    |-- pycca.exe
    `-- tack.exe
```

The executables may be found under the `tools` directory. The tools are single file executables with no external dependencies. Installation means that you copy the file somewhere in your execution path. Removal is as simple as deleting the file. There is no separate installer program. Make sure of your installation by executing `pycca -version` in a command line window (of your favorite command interpreter shell).

## Pycca

The `pycca` executable is included in this distribution under the `tools` directory. The source code for pycca is available from the either of the following repositories:

- Model Realization pycca

- chiselapp pycca

These repositories are fossil repositories. If you wish to explore the source code deeply, it is recommended that you install `fossil` and clone the repository. The fossil web site has all the necessary instructions.

The source code to the model execution domain that is used with pycca is also available from the same repositories.

- Model Realization ST/MX
- chiselapp ST/MX

### Tack

The `tack` program is a companion test harness generator for `pycca` generated domains. `Tack` is not discussed in the *Models to Code* book, but is used here to make the build of the domain executables easier. The `tack` executable is include in this distribution.

The complete source code and description of `tack` is available in the following repositories:

- Model Realization tack
- chiselapp tack

The primary use for `tack` here is to provide stubs for the external operations. The executables build using `tack` provide a local host socket interface that can be used to drive the harnessed domain. This is described in the `tack` manual, but we do not provide any examples here.

### Atangle

The `atangle` program is used to extract program "chunks" from the literate program source files. In our case, we use it to extract `pycca` source statements from the surrounding literate program description. Source code and documentation is available from the following repositories:

- Model Realization atangle
- chiselapp atangle

## Examples

The included examples have been built to run in a POSIX environment. Executables for Linux, MacOSX and Windows using Cygwin[1] are included in this distribution. In addition, all intermediate files for the examples are also included. This includes, for example, the "C" files generated from `pycca`. The only intermediary files *not* included are compiler generated object files. See below for more information about rebuilding the example programs.

### Libtack

The `libtack` directory contains both the files for the ST/MX run-time and those needed by the `tack` test harness generator. The contents of the `libtack` directory is shown below. All the example builds use the same `libtack` source files.

[1] It is necessary to have the cygwin libraries available to run the pre-built executable. There is no native Windows build included in the distribution

```
libtack
|-- harness.c
|-- harness.h
|-- libtack.a
|-- Makefile
|-- mechs.c
|-- mechs.h
|-- mechsIO.c
|-- mechsIO.h
|-- pycca_portal.c
|-- pycca_portal.h
`-- README.txt
```

### Example Source Directories

The layout of files for Chapter 4 is shown below. It is typical of the layout used for all other chapters.

```
chapter_04
|-- code
|   |-- common
|   |   |-- atcharness.tack
|   |   `-- platform.c
|   |-- linux
|   |   |-- atc
|   |   |-- atcharness.c
|   |   |-- atcharness.h
|   |   |-- atcharness.ral
|   |   |-- atctrl.c
|   |   |-- atctrl.h
|   |   |-- atctrl.pycca
|   |   |-- atctrl.ral
|   |   `-- Makefile
|   |-- macosx
|   |   |-- atc
|   |   |-- atcharness.c
|   |   |-- atcharness.h
|   |   |-- atctrl.c
|   |   |-- atctrl.h
|   |   |-- atctrl.pycca
|   |   `-- Makefile
|   `-- windows
|       |-- atc.exe
|       |-- atcharness.c
|       |-- atcharness.h
|       |-- atctrl.c
|       |-- atctrl.h
|       |-- atctrl.pycca
|       `-- Makefile
`-- model
    |-- atc-states.pdf
    |-- atc-states.uxf
    |-- atctrl.aweb
    |-- atctrl.pdf
    |-- class-diagram.pdf
    |-- class-diagram.uxf
```

```
|-- docinfo.xml
|-- duty-station-states.pdf
|-- duty-station-states.uxf
`-- Makefile
```

The `code` subdirectory is divided between the three supported platforms and a set of files common to all platforms. Each plaform directory, *e.g.* linux, contains an executable (in this case, `atc`) and the intermediary files used to build the executable. So, the `atctrl.pycca` file is the `pycca` source for the Air Traffic Control domain. It was obtained by extracting it from the domain workbook literate program source using `atangle`. The `atctrl.c` and `atctrl.h` files are the source and header file generated by `pycca`. Each platform directory contains all the necessary intermediary files and they those files were build using the platform specific version of the tool. The `Makefile` contain the build commands.

The `model` directory contains the literate program document for the example model. The PDF file there, `atctrl.pdf` in this case, contains the full xUML model and `pycca` implementation. The other files are typically graphics of the model. See below for how the PDF for the literate program document is built. Files with `.uxf` suffix are UMLet drawing files.

## Building the Code

Although executables and intermediary files are provided in this distribution, readers may wish to perform their build of the example programs. The custom tools specific to the translation technique are provided in the tools directory.

In a Linux environment, modern linux distributions include all the compilers and build tools needed to create the executable. For MacOSX, it is necessary to install `XCode`. `XCode` provides the necessary "C" compiler and `make`. For Windows, it is necessary to install Cygwin. The 32-bit version is recommended as it has a more complete set of packages. The GNU compiler, `gcc`, and `make` are available from the cygwin repositories. A native Windows build using Microsoft tools is not supported in this distribution.

## Building the Documents

The domain workbook documents are valid asciidoc documents. For a modern Linux distribution, `asciidoc` is available in the distribution repositories. For MacOSX, `asciidoc` is available from one of the ports of GNU tools to MacOS, *e.g.* MacPorts. For Windows, `asciidoc` is available from the cygwin repositories.

Note that the `asciidoc` tool chain is quite long and complicated. We build the domain workbook PDF's using docbook output and `dblatex` as the backend formatter. This introduces a considerable number of dependencies to the PDF production. Some patience may be required to get all the pieces in place in the MacOS and Cygwin environments. For this reason, the formatted PDF's are included.