Table of Contents

Introduction

Welcome to ChapMan

ChapMan is an application development environment and source code organizer for "Visual Smalltalk 3.x"

It is distributed both as a commercial product and as a restricted shareware version to allow you risk-free testing and enables us to provide you with a full featured product at a quite reasonable price. You may try the unregistered version for 30 days. If you decide to use it beyond the trial period you must register (i.e., buy the software). See License information. For ordering information or if you have any suggestions or questions please read the Ordering & Contact information section.

If you use the program after the trial period but don't pay for it, you will force us to stop distribution of the shareware version. This will deprive you of getting a high quality development tool for such a good price and the ability to thoroughly test a product before buying it. Remember all that software you have bought and don't use, just because you couldn't find out that it won't suit your needs before you bought it!

See Installation for information how to complete the ChapMan installation and Overview and Description to get an overview of the ChapMan system. This reference contains help panels for the ChapMan window as well as general information about the product, installation, ordering and license terms.

It has been arranged so that you can read it like a book simply using the **Forward** and **Back** buttons.

License and Distribution Agreement

This text describes the only terms and conditions under which Parox GmbH ("PAROX") permits distribution and usage of the ChapMan product ("PROGRAM").

This software is copyright 1995 by PAROX. All rights reserved except those specifically granted by this agreement. PAROX reserves the right to revoke or change these rights without prior notice.

Non-registered users are granted a limited license to use PROGRAM for a trial period of 30 days for the purpose of determining whether PROGRAM is suitable for their needs. Use of PROGRAM, except for this limited purpose, requires registration. Use of non-registered copies of PROGRAM by any person, business, corporation, governmental agency or other entity institution after this trial period is strictly forbidden.

Registration grants a user the right to use PROGRAM only on a single computer; a registered user may use the program on a different computer, but may not use the program on more than one computer at the same time.

No one may modify PROGRAM in any way, including but not limited to decompiling, disassembling or otherwise reverse engineering the program.

Anyone may distribute the entire PROGRAM package in an unmodified and complete form for a fee as long as the price charged for the disk containing PROGRAM does not exceed the equivalent of US $10.

With this single exception, the sale of PROGRAM or its parts for profit, either alone or together with other software or hardware, requires a licensing agreement providing for royalty payments.   Please write for terms.

DISCLAIMER:

This software is provided on an "as is" basis without warranty of any kind, expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The person using the software bears all risk as to the quality and performance of the software. PAROX will not be liable for any special, incidental, consequential, indirect or similar damages due to loss of data or any other reason, even if the author or an agent of the author has been advised of the possibility of such damages. In no event shall PAROX's liability for any damages ever exceed the price paid for the license to use the software, regardless of the form of the claim.

Gerichtstand ist der Firmensitz der Parox GmbH.

Ordering & Contact Information

You may use ChapMan for 30 days to evaluate its performance and its suitability for your needs. After this period you are not allowed use the product without registration (buying it).

ChapMan is a development environment offering more features that you usually get in environments costing considerably more than ChapMan.

Registering provides you with the following:

1. Full source code of the ChapMan system. We believe that the availability of source code is a one of the keys to the success of Smalltalk. Having the source code allows for customization and easy extension to improve the productivity of all tools.

2. No regular delays for copyright notices as in the non-registered version.

3. No restrictions as in the non-registered version, i.e., class documentation and creation of IPF help panels from Smalltalk menu source code.

4. Free support and upgrades to all 2.x versions.

Compare yourself!

In the USA
You may buy ChapMan from Parox directly or from The Smalltalk Store, which currently has the exclusive distribution rights for the US. The suggested retail price is $199. Please contact

The Smalltalk Store
405 El Camino Real, #106
Menlo Park, CA 94025
Tel. (415) 854-5535, Fax. (415) 854-2557
Email: info@smalltalk.com or 75046.3160@compuserve.com

Outside the USA
Outside the US, ChapMan is distributed directly by Parox (please inquire if you are interested in ChapMan distribution).

Price:
ChapMan (current version is 1.4) is DM 299 (approx. US $199). Please inquire for site-licensing or bulk discounts. See also Special Offer

Shipping and handling:
Germany DM 10, Europe DM 20, Worldwide (Airmail) DM 35, US $20

Value Added Tax (VAT) (Europe only)
If you order ChapMan in the European Union you must add 15% VAT (Mehrwertsteuer in Germany) to the product and handling charges. If you are outside Germany and you have a VAT registration number you don't need to pay the tax, but supply us with this registration number instead.

Payment:
Eurochecks, VISA and Diner's Club credit cards.

Bank transfer in advance to:
Postgiroamt Dortmund
BLZ: 440 100 46
Konto: 59 68-464

How to contact us:

Parox GmbH
attn: Carsten Härle
Drechslerweg 40
D-48161 Münster
Germany

Fax: +49 2534-1780
CompuServe: 100045,1257

Select Order Form for a ready-to-print form for ordering ChapMan.

Special Offer & Contributions

Bonus for Contributions

All registered users are invited to make suggestions or even implement new features to the ChapMan system. Completely implemented features are ranked for their value and if delivered with the next version the contributor

- is credited in the ChapMan documentation
- will receive several or even unlimited upgrades to his version of ChapMan, based on the ranking for his contribution.

Order Form

Fax this form to +49 2534 1780 or mail it to the following address:

Parox GmbH
Drechslerweg 40
D-48161 Münster
Germany

Order Information:

Quantity	Product	Price	Total
	ChapMan 2.x for VST 3.0 OS/2 AND Win32		
1	Shipping and Handling		
	Total		

Customer information:

Name: ____ Date: ____________

Company: ______________________________________ Dept: ______________

Address: ____

Country: ____

Tel: _________________________ Fax: ____________________________

Email: ____

Payment:

__ Advance bank transfer to:
Postgiroamt Dortmund, Germany
BLZ: 440 100 46, Konto: 59 68-464

__ Eurocheck enclosed

__ Diners Club __ VISA

Card Number: ______________________ Expiration Date: _____________

Card Holder's Signature REQUIRED: __________________________________

Comments:

Where did you get your evaluation copy from: __________________________

What would you like to see in the next version: _______________________

____

Registration Form

Email this form CompuServe 100045,1257,
fax it to +49 2534 1780 or mail it to the following address:

Parox GmbH
Drechslerweg 40
D-48161 Münster
Germany

Customer information:

Name: ____ Date: ____________

Company: ______________________________________ Dept: ______________

Address: ____

Country: ____

Tel: _________________________ Fax: ____________________________

Your preliminary registration number: __________________________________

Comments:

Where did you buy your copy of ChapMan: ______________________________

____

What would you like to see in the next version: _______________________

____

Installation

ChapMan distribution files:

readme
    Read this first.

ChapInst.Cls
    ChapMan Installation Program

ChapBas.st
    ChapMan Base System File In

ChapO20.inf, ChapO20.hlp, ChapW20w.hlp
    Online help and documentation for OS/2 and Windows.

ChapO20.SLL, ChapO20.SML, ChapW20.SLL, ChapWO20.SML
    ChapMan base system code in SLL format for OS/2 and Windows.
    ChapO20.SML is only included in the full version.

ChapO20.SLL, ChapW20.SLL
    ChapMan Registration System SLL for OS/2 and Windows.

ChapO20.DLL, ChapW20.DLL
    ChapMan resource DLLs for OS/2 and Windows.

*.cha, *.cls
    Application definition files and application source files.

Installation:

1. Just file in the file **ChapInst.Cls** and see the README file for further instructions.

2. If you bought the full version of ChapMan enter your registration number by opening a *ChapMan Application Browser* from the *Smalltalk* menu select *Help/Product Information/Register*. If you have a negative (i.e., preliminary) registration number, your copy is not registered directly with Parox. You should register directly with Parox to be eligible for free upgrades on the same major version number. Select Registration Form for a ready-to-print registration form.

3. Open the Options/ChapMan: General dialog to set the programmer id.

Overview and Description

See the following sections for more information on current and future features.

- Features
- Things to come
- Revision History
- Changes to Digitalk's environment

The following sections give you a summary of the features as well as a general description how the ChapMan system works.

- Applications
- Class Documentation
- Method Categories
- Method Histories
- Smalltalk Lint
- Tips and Frequently asked Questions

Features

The whole system has full drag/drop support, context sensitive online help and multiple selection list boxes, to greatly improve the usability of all tools.

ChapMan Application Browser

- Browse and revert to any old method definitions.
- Source code organization into a hierarchy of applications and subapplications.
- Source documentation including semi-automatic change logs for each method, class comments and instance variable descriptions.
- Consistency checks such as searching for undefined message sends and unused methods.
- Various reports including a listing of changes to methods done after a certain date and a summary of source code locations.
- Create your own hierarchy of classes with complete class protocol descriptions like in the STV documentation. Production of ready to print RTF files.
- Method categories.
- Many general goodies like browsing implementors, senders of symbols and users of global or pool dictionary variables by just marking a symbol in any text pane.
- Vastly improved Class Hierarchy Browser with dozens of useful goodies (including rename class, change superclass, change class of a method and automatic creation of access methods for instance variables).
- Automatic Smalltalk LINT checks for unimplemented messages after each method save.

ChapMan Source Browser

- Browsing of arbitrary Smalltalk source files, including ParcPlace source files.

- Intelligent chunk type detection (e.g., "define class", "open window", "saved image", "method definition" etc.).

- Conflict detection with current system definitions.

- Display the latest versions of a definition only. Useful for browsing change.log files.

- Easy marking, flexible pattern matching and filtering to allow easy selection of a specific set of changes.

- Integration of the source code browser with the application management allowing selective addition of definitions from a chunk file to an application.

- Install selected changes with or without prior modification.

- Resolve class name conflicts by renaming classes in the browsed changes file.

ChapMan Library Builder

Create Smalltalk/V SLL files from applications.

ChapMan IPF Browser

- Create and maintain *Information Presentation Facility* files for documentation and online help.

- Parse menu definitions from the source code and automatically create IPF headings from them.

Changes to Digitalk's environment

Some changes have been made to the original Smalltalk/V environment. Some are just improvements that are useful in general, some others have been made to achieve a better integration of ChapMan into the system. The following changes have been made:

- Three new items in the *Smalltalk* menu of all windows. These allow the user to open the corresponding ChapMan windows. They are **Browse Application**, **Browse Changes** and **Edit IPF Files**.

- For a description of the new **Senders**, **Implementors** and **Format Comment** entries in the Edit menu select the particular keyword.

- The *Smalltalk*-Menu in the Transcript window has some tools which are usually evaluated as Smalltalk expressions.

- There is an enhanced *OrderedCollectionInspector* that allows the addition of elements to the inspected collection.

- The *Smalltalk*-Menu of the *System Transcript* has now some extra choices for frequently used actions.

Revision History

Version 2.0 (22nd June 1995)

- Complete restructuring to meet Visual Smalltalk requirements.
- One version for Windows AND OS/2.
- Improved Smalltalk-Lint feature now finds even more undefined message sends.
- Multi-platform applications.
- New menu items: Classes/References
- Organization only applications to organize system code.
- Improved creation of class documentation and various formats including RTF.
- Browse and file in ParcPlace Smalltalk files.
- Method categories.
- Improved user interface with more dialog boxes, multiple selection listboxes in almost all browsers, full Drag/Drop support.
- Improved OrderedCollectionInspector.
- Team/V compatibility.
- New installation program for easy installation.

Version 1.4 (17 October 1994)

- Browse and revert to any history version of a method.
- Bug fixes: Default Variables could not be set; check for methods sending #halt didn't work.

- Perform AutoSTLint in Method Browsers also.
- New "Application File In Options" dialog box.

Version 1.3 (13th September 1994)

- Free maintenance update.
- Bug in the registration system, made the Shareware-Version 1.2 unusable.
- Some other minor bug fixes.
- Upgrade to versions with the same major version number is now free of charge.

Version 1.2 (10th September 1994)

- Improved code documentation features. Create your own *Encyclopedia of Classes* like in the Smalltalk/V manual.
- New Reports: All methods changed after a certain date with their changes, Source Code Location report for methods and classes, Search Source for patterns, and others.
- The *ChapMan Source Browser* (renamed from ChapMan Change Browser) is now used as an enhanced Method Browser.
- Bug fixes: renaming classes caused loss of the class comment, working with marked changes and 'display recent versions' caused incorrect selection in the changes list in the *ChapMan Source Browser*, problem with automatic Get/Set method creation, any other small ones.
- More robust logging mechanism for ChapMan variables and class comments in the change.log - won't cause walkbacks if the class or application doesn't exist during reinstallation.
- Filters to separate public and private messages in the ChApplicationBrowser.

- Many menu choices referring to classes or methods now allow for multiple selections to be supplied from a dialog box.

- Option to remove all classes and methods of an application from the system when removing the application.

- *Application/File In Applications* now optionally files in the Smalltalk source also. This allows for easy installation and uninstallation of applications if used in conjunction with the previous point.

- Automatic Smalltalk LINT checks for unimplemented messages after each method save.

- Improved selection dialog box: updates its list while you enter characters and wildcard patterns.

- Many more improvements in the user interface to increase flexibility.

Version 1.1 (11th May 1994)

- All system classes can now be browsed in the *ChapMan Application Browser*. For that reason support for Digitalk's CHB has been dropped. You should now use only the CAB.

- New *ChapMan IPF Browser* for developing IPF documents and online help.

- Various features added to the CAB: change the superclass of a class, change the class of a method, move classes and methods to other applications, direct execution of the selected method, file out the code for the selected class, and other minor improvements.

Version 1.0 (28th Feb 1994)

First release.

Things to come

The following features are currently under development:

- Improved ST-Lint features, such as checking for private methods.
- Semiautomatic spelling corrector.
- Improved Library Builder.
- Define more than one source for a method to facilitate the creation of different versions of your application for different purposes.

And of course: All your suggestions are welcome. For your contributions see also Special Offer.

Applications

A ChapMan application is a collection of classes and methods which form a unit in some sense. Each non-system method should be a member of some application to ensure that your source code can be controlled by ChapMan.

Applications can have subapplications in the sense that any application needs all its subapplications for its definition. For example:

```
ChapMan Integration
 ChapMan
  General Tools
```

The *ChapMan Integration* application contains code to integrate ChapMan into the system. Therefore it is a superapplication of the *ChapMan* application. The *ChapMan* application is a super application of *General Tools* because the ChapMan implementation needs some classes and methods that sare defined in the *General Tools* application.

There are two special applications called **<SYSTEM>** and **<USER>**. The first one contains all classes and methods of the system, so select this one to use the Application Browser as an enhanced *Class Hierarchy Browser*. By convention all Organization Only Application should be subapplication of this application, whereas all real user applications should be subapplications of the **<USER>** application but this is not enforced (remember that you can simply change the super application by using drag/drop).

Organization Only Applications

An organization only application is just for organizing system code into applications. Such an application does't have a source file and methods or classes in it are not considered to really belong to the application. This is important for the check if all user methods are assigned to applications, or the check for methods contained in multiple applications. Organization only applications are excluded from these checks, e.g., ChapMan will not complain about a system method to be part of an organization-only application.

You may change this property in the Application Settings Dialog.

Multiplatform Applications

ChapMan offers the possibility to create applications that file in on multiple platforms, by storing the application in separate files for each platform.

You may create such an application by specifying a question mark ('?') anywhere in the file name of the application. This question mark will be replaced by a platform-specific letter during file-in and file-out. Currently these letters are

'o'
: for an OS/2 System

'w'
: for a Windows Win32 System.

For example consider the following application hierarchy:

```
Userinterface Extensions (File name: 'UIExten?')
  Userinterface Extensions Portable (File name: 'UIExPort')
```

The application has been split into a platform-independent application (the application named 'Userinterface Extensions Portable') which well be saved into the 'UIExPort.Cha' and 'UIExPort.Cls' files, and a platform-dependent part called 'Userinterface Extensions' which will be saved into the files 'UIExteno.Cls' and 'UIExteno.Cha' on an OS/2 system and 'UIExtenw.Cls' and 'UIExtenw.Cha' on a Windows system.

The advantage over defining separate applications called 'Userinterface Extensions OS/2' and 'Userinterface Extensions Win' is that with the former method, the application hierarchy keeps the same on all platforms, and an application containing platform-specific subapplications still needs only one source file to file in correctly, as ChapMan selects the files for platform-specific applications automatically.

The same method can be applied to Smalltalk Link Libraries (SLLs). Giving a library a name containing a question mark instructs ChapMan to replace the individual platform letter when it build the library.

Class Documentation

ChapMan provides many tools for effortless, high quality class documentations. You may create your own *Encyclopedia of Classes* in only a few seconds. Various output formats are available including ASCII and RTF (Rich Text Format), which may be printed and processed by almost all popular word processors.

Creating documentations with ChapMan is so easy, because much information, such as method comments, is already available and other information such   as class and variable comments can be supplied easily during the development phase. Integrating the documentation and development phase saves a lot of time and leads to much more accurate results than doing it in two separate steps.

To create the documentation for a class, ChapMan takes the method comments, the assigned method categories, the class comments and the instance and class variable descriptions and produces neatly formatted text from it. See Documentation Example for an example of such a documentation in ASCII format. See Class and Variable Comments for more information how to set up these descriptions.

Class Documentation Example

This is an example of a class documentation for the class *ChFormatter* in ASCII format.

```
ChFormatter

This is an abstract class which provides the public protocol to
output text in a formatted way. New paragraph indents
(leftIndent,
firstLineIndent, rightIndent) must occur before any text for the
current
paragraph has been written to the receiver. The settings take
effect for
the current paragraph.

22.05.95 ch Creation. Copyright (c) 1995.


Inherits from:
    Object

Inherited by:
    ChAsciiFormatter ChChaFileFormatter ChRtfFormatter
    ChRtfAmiProFormatter

Pool dictionaries:
    NONE

Named instance variables:

    braces
        <Integer>. Holds a count of the currently open
        environments.

    firstLineIndent
        <Integer>. The indention of the current paragraph in
        native format of the concrete subclass.

    leftIndent
        <Integer>. The left indention of the current paragraph
        in native format the concrete subclass.

    rightIndent
        <Integer>. The left indention of the current paragraph
        in native format the concrete subclass.
    stream
        <Stream>. The output stream.


Class variables:
    Dependents (from class Object)
    EventHandlers (from class Object)
    RecursionInError (from class Object)
    RecursiveSet (from class Object)


Class methods (Public-*):

    new
        Answer an instance of the receiver.


Instance methods (Public-*):

    beginBold
        Start a bold section.

    beginDocument
        Start the document and write any important headers to
        the stream of the receiver.

    beginFontSize: anInteger
        Begin a font size environment with font size <anInteger>
        points.

    beginItalic
        Start an italic section.

    bullet
        Write a bullet to the output stream.

    endBold
        Start a bold section.

    endDocument
        End the document and write any important trailers to the
        stream of the receiver.

    endFontSize
        End the font size environment.

    endItalic
        Start an italic section.

    firstLineIndent
        Answer the value of firstLineIndent.

    leftIndent
        Answer the value of leftIndent.

    page
        Write a page break to the receiver.

    paragraph
        Make a new paragraph.

    rightIndent
        Answer the value of rightIndent.

    stream
        Answer the value of stream.

    stream: aValue
        Set the value of stream.

    text: aString
        Write a the text <aString> to the receiver stream.
        Ignore white space in a String and insert a single space
        between word.

    textLine: aString
        Output <aString> without lines breaks.
```

Class and Variable Comments

The class comment for a class can be edit by just selecting the class in the *Application Browser* and entering the text after the '@' character in the text pane.

The instance variable and class variable descriptions can be supplied after line just containing '@' character in the class comment and must have a special format to be correctly parsed by the class documentation system.

Any text up to the first line starting with a dot will be ignored. Variable descriptions start with a dot in the beginning of a new line followed by the variable name, an optional ':' character and the variable description which may by any text up to the next line starting with a dot. More formally:

```
<variable_description_section>
    : <text> <variable_description>*
    | <variable_description>* ;

<variable_description>
    : '.' <variable name> [':'] <text> ;

<text>
    : <any characters up to a line beginning with a dot> ;
```

Here is an example for such a list:

```
Variable description:

.braces : <Integer>. Holds a count of the currently
    open environments.

.firstLineIndent : <Integer>. The indention of the
    current paragraph in native format of the concrete subclass.

.leftIndent : <Integer>. The left indention of the
    current paragraph in native format the concrete subclass.

.rightIndent : <Integer>. The left indention of the
    current paragraph in native format the concrete subclass.

.stream : <Stream>. The output stream.
```

Method Categories

In ChapMan methods can be organized in *method categories* to allow faster access to certain groups of methods, to select certain groups of methods for documentation purposes and to distinguish between public and private methods of a class.

The name of a method category is a string not containing any whitespace. For maximum compatibility and ease access, method categories are simply stored in the first method comment of each method. Two formats are currently supported:

- Separate the category name and actual method comment by any whitespace followed by a dash. Formally:

  ```
  <method_comment> : <category_name> <whitespace> '-'
  <actual_comment>
  ```

  For example:

  ```
  createView
          "Private-GUI - Create the views for the receiver."
      | temp |
      ...
  ```

- Enclose the category name in parenthesis. More formally:

  ```
  <method_comment> : '(' <category_name> ')' <actual_comment>
  ```

  For example:

  ```
  createView
          "(Private-GUI) Create the views for the receiver."
      | temp |
      ...
  ```

  There are five special categories provided by ChapMan that should not be used as user category names with one exception. The category named **Private** may be used like any other category. It only has a special meaning to ChapMan. The special categories are:

All

This category contains all methods of the class.

Public
Contains all methods with no specific category assigned.

Public-*
Contains all public methods, which are by definition all methods whose category name does not begin with the word *Private* (case insensitive).

Private
Contained all methods whose category name is exactly *Private*.

Private-*
Contains all private methods, which are by definition all methods whose category name begins with the word *Private* (case insensitive).

Method Histories

ChapMan automatically saves any old version of methods you change in the system, so you can always revert to any of these versions if you discover that the changes you made are not satisfactory or browse old versions if you need access to prior implementation details.

ChapMan just saves the source code location of the old definitions but not the actual source. This means that the file containing the source code of the old version must be present to be able to access the source code. Most commonly the source file will be system *change.log*. You will have access to all old method versions this file, but if you compress your *change.log*, ChapMan purges its method history and you cannot access old definitions any more. So be careful when you compress your changes.

Smalltalk Lint

The Smalltalk Lint is a unique feature to ChapMan, which allows to you catch typing errors and undefined methods during development instead of during testing or even never.

It simply checks, whether all message sends occurring in a method are defined as methods anywhere in the system. If it finds symbols without in implementing method, a method browser is opened with the offending method and message selectors. You can then simply select one if these entries and the offending message send will automatically highlighted in the browser and you can decide whether is must be corrected or not. This check can be either performed after each method save in a code browser or separately on a specific class or application.

ChapMan assumes that every symbol literal in a method is a message send, an assumption which is not necessarily true. You can exclude certain symbols from being listed as undefined message sends on three levels:

- o System wide (see Options/ChapMan:Symbols/Libraries)
- o Locally for an Application (see Application/Settings/Properties)
- o Locally for a class (see Class/Settings)

Consider for example the following method for the class *ChApplication*:

```
printOnNew: aStream
        "Append the ascii representation of the receiver to
<aStream>."
    aStream nextputAll: 'App{', name, '}'.
    self name ~= '<SYSTEM>' ivTrue: [
        super pintString].
```

If you save this method in an *Application Browser* with *Automatic ST-Lint enabled* you will get a *Method Browser* after a few seconds having the following entries:

```
ChApplication>>printOnNew: (ivTrue:)
ChApplication>>printOnNew: (nextputAll:)
ChApplication>>printOnNew: (pintString)
```

Just select any entry and correct the spelling errors.

Multitasking

ChapMan performs several actions in the background such as checking consistency, loading source files and optionally the automatic Smalltalk-Lint. During such actions you can simply continue with your work and don't have to wait for the action to be finished. E.g., if you open a *Source Browser* on a fairly large *change.log*, loading this file can take up to several minutes. Just continue with your work and see the *Source Browser* pop up after the loading is finished.

Tips and Frequently Asked Questions

Question: How can I set up default templates for all applications?

Answer: Make sure that no application is selected (e.g., by opening a new Application Browser) and select *Application/Settings/Edit Application Templates*. These variables become default values for all applications that have a <*default*> entry in their text variable.

Question: I filed in several files and want to know which methods have been installed and add them to certain applications.

Answer: Save your image and file in all the files you like. Then open a *Source Browser* and selected the change.log as the file in the dialog box. Answer *Yes* to question if you want to browse the file from the last image save only. After a while, the browser window opens with all method and class definitions installed since the last image save. You may now drag certain methods or classes to an application or use   the Change/Add to Application or the corresponding choice in the All Displayed-Menu to add methods to an application.

Question: How can figure out that methods do not belong to an application yet?

Answer: Use the Application/Report/Consistency choice for various consistency checks including the desired one.

ChapMan Windows

The main windows of the ChapMan system are

- ChapMan Application Browser
- ChapMan Source Browser
- ChapMan Library Builder
- ChapMan IPF Browser
- ChapMan Template Browser

ChapMan windows provide extensive support for drag/drop and much effort has been taking to design these operations as natural as possible. Many tasks that were tedious to handle in old versions are considerable simplified using drag/drop. It is therefore strongly recommended that you consult the drag/drop section for every pane in ChapMan windows.

ChapMan Application Browser

ChapMan Application Browser

The *ChapMan Application Browser* window is the central part of ChapMan. Within this window, you manage all your applications and create reports.

See Introduction for important general information.

See one of the following topics to get further information on the ChapMan Application Browser Window.

- Window Description
- Menu Description
- Dialog Window Description
- Key Assignments

Window Description

The following pages describe all panes in a ChapMan window.

- Title bar
- Application list pane
- Classes list pane
- Method Categories
- Variables list pane
- Methods list pane
- Text pane

Title bar

The title bar of a ChapMan window reflects various selections made in the window.

- ChapMan Applications is displayed if nothing is selected.
- ChapMan Applications | XXX is displayed if the application with name 'XXX' is selected.

Application List Pane

The application list appears in the upper-left pane. In this pane, the names of all the applications in the system are presented in a hierarchical order. Subapplications are applications that are required by their parent application and they are indented in each level. An individual application can be a subapplication of many other applications. See Add Subapplication.

Three dots following an application's name indicate the existence of hidden subapplications, which can be displayed by double clicking on this application. For example:

```
ChapMan Integration...
```

ChapMan Integration has subapplications that are currently hidden.

Application List Drag/Drop

The application pane supports Drag/Drop as a source for applications and a target for applications, classes and methods. Move operations are the default if an object is dragged with the right mouse button and no key is pressed, whereas holding the **Ctrl** key causes a copy operation to take place. The following actions are taken depending on the type of the dropped object:

Application
: The dragged application is added as a subapplication of the target application. Both copy and move operations are supported but circular dependencies are not.

Class
: The dragged class is added to the target application. Both move and copy operations are supported. The drag may be initiated from the class list pane in the *Application Browser* or any other class pane supporting drag/drop (e.g., *Class Hierarchy Browser*). Dragging a class from the class list pane in a *Application Browser* causes the *owned/required* state of the class and all its methods in the original application to be transferred to the target application.

Methods
: The dragged methods are added to the target application. Both move and copy operations are supported. The drag may be initiated from any method list pane in a browser supporting drag/drop (e.g., *Method Browser*, *Class Hierarchy Browser*).

Class List Pane

The classes list pane appears just below the application list pane and displays the classes which belong to or are required by the selected application.

Each class is indented with as many characters as it has super classes. Blanks indicate that the class is a direct subclass of the class in the previous line. Dots indicate that the class has as many superclasses as there are dots but the class is not a direct subclass of the class in the precious line. If a class has three dots following its name, there are hidden subclasses for it in the application. For example:

```
....String
     DoubleByteString...
     Symbol
```

The four dots before *String* indicate that its four superclasses are not contained in the application. The spaces in front of *DoubleByteString* and *Symbol* indicate that these are direct subclasses of *String*. Finally the three dots after *DoubleByteString* indicate that this class has hidden subclasses in the application.

Class List Drag/Drop

The class pane supports Drag/Drop as a source for classes and a target for both classes and methods. Move operations are the default if an object is dragged with the right mouse button and no key is pressed, whereas holding the **Ctrl** key causes a copy operation to take place. The following actions are taken depending on the type of the dropped object:

Class

Dropping a class on a target class causes the dragged class be recompiled as a subclass of the original class. Every operation is considered to be a move operation.

Methods

Dropping methods on the target class causes the methods to be compiled in the target class. Both copy and move operations are supported, and care is taking in the first case not to delete original methods if the compilation in the new class fails.

Category List

The Method Category list pane appears to the right of the classes pane and displays all method categories for the selected class plus some special categories.

The special categories are:

All
All methods.

Public
All public methods that do not have a special category assigned.

Public-*
All public methods, i.e., methods whose category name does not begin with the word *private*.

Private
Methods in the category with name *Private*

Private-*
All private methods, i.e., methods whose category name begins with the word *private*.

Categories are assigned to methods in either of the following ways

- Writing the category name, a space and a dash into the beginning of the first method comment.

```
createView
        "Private-GUI - Create the views for the receiver."
```

- Writing the category name in parenthesis into the beginning of the first method comment.

```
createView
        "(Private-GUI) Create the views for the receiver."
```

Category names may be composed of any characters but white space. To change the category of several methods just select these methods in the method pane and drag them onto the target category in the method category pane.

Category List Drag/Drop

The category list pane supports Drag/Drop a target for methods. Dropping methods on a target category causes this method to be moved to this category.

Variable Drop Down List

The variables list pane at the right top of the browser window and displayed all variables for the selected class. Refer to the Digitalk manual for further description.

In the same line you find a second drop down list which let you select how the variable filter for the selected instance of class variable should work. These filters are the same as in the e.g. in the ClassHierarchyBrowser.

Method List Pane

The method list pane is the rightmost pane of the top half of the browser. It contains a list of instance or class methods in the selected application. All filters (i.e., the Variables list pane and the method category filter) are applied before display.

Method List Drag/Drop

The method list pane supports Drag/Drop as a source and target for one single or multiple methods. Move operations are the default if an object is dragged with the right mouse button and no key is pressed, whereas holding the **Ctrl** key causes a copy to be the default operation.

Dropping methods onto this pane is the same as dropping them on the selected class, i.e., the dragged methods are compiled in the new class and in case of a move operation, they are removed from the original class.

Text pane

The text pane appears as the large bottom pane of the window. In this pane you can edit code for Smalltalk methods and class definitions. The contents of the text pane depend on the selections in the other panes as well as on certain menu selections.

Menu Description

The ChapMan Application Browser window offers the following menus:

- Menu: Edit
- Menu: Smalltalk
- Menu: Application
- Menu: Classes
- Menu: Methods

Smalltalk Menu

The choices in this menu are well documented in the Smalltalk user manual from Digitalk, apart from two new options:

- Implementors
- Senders

Application Browser

Open a new ChapMan Application Browser.

Source Browser

Open a file selection dialog where you can select a Smalltalk chunk file to browse. Then open a ChapMan Source Browser on that file. The parsing of the source file will be done in the background.

If you specify the system *change.log* as the file you will be asked if you want to browse only the changes made since the last saved image event. This is particularly useful for crash recovery.

IPF Browser

Open a ChapMan IPF Browser window.

Edit Menu

The choices in this menu are well documented in the Smalltalk user manual from Digitalk, apart from the new option:

- Format Comment

Save

Save the text in the text pane. If the text is a method definition and the *Method Comment Template* is not empty in the selected application, you may be asked further questions. See Method Comment Template. If the Automatic ST-Lint option is enabled, a *ChapMan Source Browser* window with warnings may pop up after some seconds.

The text popup menu has one additional choice: **Save w/o Lint**. Use this choice to prevent the execution of the Smalltalk Lint for this method save.

Save w/o Lint

This is the same as the Save-Choice, but the automatic Smalltalk Lint will not be performed. This is useful for example, if you add some new choices in a menu definition method, and you are going the define these new method selectors immediately. In this case you don't the window saying that some message sends are not defined.

Format Comment

Take the first string enclosed in double quotation marks (") and format it to the width set in the config variable 'CommentWidth' and indented with two tab characters from the left margin.

This is useful for formatting Smalltalk method comments.

Implementors

Take the currently selected text as a method selector and open a method browser on the implementors of this method.

Remove all separators and special characters (like parentheses) from the string first, so it usually suffices to use a double click for symbol selection even though this includes more than the actual method selector.

Prompt for a method selector if nothing is selected in the text.

Senders

Take the currently selected text as a method selector, global or pool dictionary variable name and open a method browser on the methods sending this message respectively using this variable.

Remove all separators and special characters (like parentheses) from the string first, so it usually suffices to use a double click for symbol selection even though this includes more than the actual method selector.

Prompt for a name if nothing is selected in the text.

Application Menu

The following choices are available from the application menu.

- New Application...
- History...
- Browse all Methods...
- Report
- Update
- Collapse All
- Expand All
- Settings
- File Out Applications...
- File Out Install Code...
- File In Applications...
- Open Library Builder
- Remove

New Application...

This choice causes an *Applications Settings* dialog to popup where you can enter the name and properties of a new application. This application will be added as a subapplication to the currently selected application. If no application is selected it will be added to the application with name *<SYSTEM>* if it is an organization only application and to the application with name *<USER>* otherwise.

It is possible for an application to be a subapplication of many other applications, but circular dependencies are not allowed.

History...

Open a browser on all history versions of all methods in the selected application. See Method History for more information.

Browse all Methods...

This choice opens a method browser on all methods contained in the selected application.

Report Menu

ChapMan provides the following application reports.

- Class Documentation...
- Changed Methods Since...
- Messages without Implementors...
- Methods without Senders...
- Search Source Code...
- Class Ownership...
- Consistency...

Class Documentation...

Create an *Encyclopedia of Classes* for all classes in the selected application.
See Class Documentation for more information.

Changed Methods Since...

You are asked for a date and a report with all methods in the application whose automatic comments indicate a change after the specified date is created. The date may be specified also in a relative manner, i.e., you may type "-4" to specify the current date minus four days.

For each method in this list, the method selector, the method description and all automatic comments dated after the specified date are displayed.

Messages without Implementors...

Make a report listing all symbols occurring in methods of classes in the selected application that don't have implementing methods. You will be prompted, if you would like to include subapplications. This report will be created in a background process.

Use the Application/Settings/Config Variables. choice to control which symbols are considered to be message sends.

Methods without Senders...

Open a method browser on all methods of the selected application that have no senders from any class of the Smalltalk system. You will be prompted, if you would like to include subapplications.

Search Source Code...

You will be prompted to supply a pattern which may contain wildcard characters ('*'). The source for all the methods in the selected application is searched for matches and a *ChapMan Source Browser* is opened on the resulting collection.

Class Ownership...

Display all applications with a list of the required and owned classes.

Consistency...

Check whether all user methods and classes are in applications, whether some items are in more than one application, and perform other checks for consistency. All report values should be NONE.

User and system class or methods are distinguished by the library in which they reside (V.EXE has library number 0). See Options/ChapMan: Symbols/Libraries for further information how to specify which SLLs are user SLLs and which should be system SLLs. See Report: Source Code Location to get information how to obtain the SLL number of methods.

Update

Update the application display pane.

Collapse All

Collapse all subapplications for all applications.

Expand All

Expand all subapplications for all applications.

Settings Menu

This menu let you change settings for the selected application.

- Edit Application Templates...
- Application Settings...

Edit Application Templates

Open a ChapMan Template Browser on the selected application or on the default values if no application is selected.

Application Settings...

Open an application settings dialog.

File Out Applications...

Save the Smalltalk/V code (file extension ***.CLS**) and the application description (file extension ***.CHA**) of one or more applications.

If an application is selected, the dialog box will show the selected application as the default choice.

File Out Install Code...

Open a dialog where you can select one or more application from. If an application is selected, the dialog box will show the selected application and all its subapplications as the default choices.

Create an installation file (extension ***.ST**) for the selected applications. You will be prompted to supply a directory pathname where subapplication are to be installed from. Just enter any path name or leave this field blank, in which case the installation script will prompt you for a directory upon execution. You may also enter a global Smalltalk variable name prefixed with a question mark. This causes the installation script to take the string stored in this global variable (provided it exists in the system) as the default directory. In any case you will be prompted for a new directory name, if the installation script is not able to find a certain file during execution.

Currently there are not checks if the application hierarchy is properly defined, so that the created installation file may not work. This may be the case an application tries to use a class that have not been defined yet. To fix this you have to make the application defining the offending class a subapplication of the application where the error occurred. Alternatively you may use a separate class definition file created by File Out Class Definitions.

File Out Class Definitions...

Open a dialog where you can select one or more application. If an application is selected, the dialog box will show the selected application and all its subapplications as the default choices.

File out all class definitions of classes owned by the selected application.

File In Applications...

File in the application definition and optionally the source code from the application files (**CHA** and **CLS**).

Use the file dialog box to select the files you like to file in. After this a second dialog box will be opened where you can select various options. If subapplications are to be filed in, ChapMan searches these applications in the current application directory (see Global Settings Dialog) not in the directory of the root application filed in. If you want the application to be filed in from this directory you have to temporarily change the application directory to the desired one.

Open Library Builder

Open a ChapMan Library Builder on the selected application.

Remove

Remove the selected application from the current position of the application list. If the selected application does not occur as a subapplication of some other application, this deletion must be confirmed. If the deletion is confirmed, then all information contained in the application is lost.

In the latter case, you are asked if you want to remove all the classes and methods in the application from the system also.

Classes Menu

The following items are available from the **Classes** menu:

- Add (Sub)class...
- Rename Class...
- Own Class
- Import Classes...
- Report
- Browse Class
- References
- History
- File Out Class...
- Find Class...
- Find Application
- Settings...
- Remove from Application
- Delete

Add (Sub)class...

Create a new class and add it to the selected application. You will be prompted to select a superclass in a dialog box, which displays the selected class as the default choice.

Rename Class...

Rename the selected class.

If there are references to the selected class, you are prompted whether you want to browse all references to the class. This source code needs to be updated to the new class name.

Own Class

Toggle between the owner and the required state of the selected class. A check mark indicates whether the selected application already owns the selected class.

An application owning a class contains all of its instance and class methods. If you want to include only some of the methods for a class, you must turn off class ownership.

Import Classes...

Import one or more classes into the selected application. If the selected application is an *organization only*, the *owned* state is automatically set.

Report Menu

The following class reports are available.

- Super- and Subclasses
- Class Documentation...
- Changed Methods Since...
- Source Code Location
- Methods without Senders
- Methods without Local Senders
- Messages without Implementors
- Search Source Code...

Super- and Subclasses

Display a text window containing the hierarchy of superclasses for the selected class. The selected class is displayed in parenthesis.

Example: Selecting this report with the class *ChCodeBrowser* selected, leads to the following output:

```
Object
  ViewManager
    Browser
      CodeBrowser
        (ChCodeBrowser)
          ChApplicationBrowser
          ChSourceBrowser
```

Class Documentation...

Create a documentation for the selected class. The output format and some other options may be selected from the displayed Class Documentation Dialog

For more information on class documentation see Class Documentation.

Changed Methods Since...

You may supply a date and ChapMan lists all methods in the selected class whose automatic comments (those starting with an '@') indicate that they have been modified after the supplied date. You may also specify a string like "-4", which specifies today's date minus four days.

For each such method list the method selector, the method description and all automatic comments dated after the specified date. For example:

```
Report created 03.06.95 00.46.52

Changed methods in 'ChapMan Kernel' since 01.05.95 for class
ChApplication.

ChApplication class

    getApplicationDefinitionFrom: aStream
        Answer a dictionary with all key/values pairs in the
        header of the ChapMan application file.
        @04.05.95 ch: use exceptions

ChApplication

    extraSymbols
        Answer the value of extraSymbols.
        @02.05.95 ch: Creation

    fileInDefinitionFrom: aStream subApplications: doSubApps
        installSource: installSource useExisting: useExisting
        Private - Reconstruct the receiver definition from
        aStream. If installSource is true try to file in the
Smalltalk
        source code. If useExisting is true and the applications
in the
        file exists already, answer the existing application.
Answer
        the receiver or nil if operation fails.
        @02.06.95 ch: don't file in source of organizationOnly
        apps
        @24.05.95 ch: use #comment: to set the class comment
        @04.05.95 ch: use exceptions
        @04.05.95 ch: revised the processing of the text
        variables
        @03.05.95 ch: support extra symbols and organizationOnly
    fileOutCodeOn: aStream report: report
        File out the initialization, finalization code, class
        definitions and methods definitions of the receiver on
aStream in
        the correct order (i.e., that they file in properly).
        @24.05.95 ch: use the new #comment method to get the
        method comment

    fileOutDefinitionOn: aStream report: report
        File out the Application definition out on aStream. Put
        the application name, date and time, all classes and
methods
        associated with the receiver and all class descriptions.
        @22.05.95 ch: use ChFormatter classes
        @03.05.95 ch: support extra symbols and organizationOnly

    fileOutReport: report
        File out the receiver to the files with base name
        defined in the text variable 'Filename'. Append .cls for
the
        code and .cha for the definition file.
        @04.05.95 ch: don't file out code of organizationOnly
        apps
        @04.05.95 ch

    name: aString
        Set the application name to aString. Answer the receiver
        if operation was successfull otherwise nil.
        @12.05.95 ch: do nothing if <aString> is the own name
```

Source Code Location

Report where the compiled method object, the method source and the class object reside in the Smalltalk image. Also give a description of all source destination files and bound libraries. For example:

```
Report created 03.06.95 00.52.49

Class ChSystemRootApplication (Library: Image)

Format is:
<selector> (<source index>, <library name>)

Note: Methods with source index greater than 3
have the source in their library.

Source index values:
1: nil
2: a FileStream on: 'change.log'
3: a FileStream on: 'vbas3ao.sml'

Library index values:
0: Image
1: Base-Library
2: vdev3ao.sll
3: VOSW3A
4: VSLB3A

Class methods: NONE

Instance methods:
    allClassesBasic  (2,Image)
    defaultDisplayClasses  (2,Image)
    includesClass:  (2,Image)
    initialize  (2,Image)
    isOwnerOf:  (2,Image)
    methodsForClass:  (2,Image)
```

Methods without Senders

Open a method browser on all methods of the selected class in the selected application that have no senders from any class of the Smalltalk system.

Methods without Local Senders

Open a method browser on all methods of the selected class in the selected application that have no senders in the class itself.

Messages without Implementors

Create a report with symbols occurring in methods of the selected class that don't have implementing methods.

For a more powerful search with more many options select the corresponding choice on the application menu (Application: Messages without Implementors).

Search Source Code...

You will be prompted to supply a pattern which may contain wildcard characters ('*'). All the method sources for the selected class are searched for matches and a *Method Browser*

Browse Class

Select a class in the displayed dialog box or type a pattern in the entry field (case is not important and wildcard are allowed). You will see a list of matching classes as you type ahead. If only one class matches the pattern it is automatically selected and you may hit **Enter** to accept this class.

A *Class Browser* will be launched for this class.

References

Open a method browser on all methods referencing the selected class. If no class is selected a prompter is displayed and you may enter any global variable name.

History

Open a browser on all history versions of all methods in the selected application and class. See Method History for more information.

File Out Class...

You will be prompted to supply a file name where the source code of the selected class should be written to.

Find Class...

Select a class in the displayed dialog box or type a pattern in the entry field (case is not important and wildcard are allowed). You will see a list of matching classes as you type ahead. If only one class matches the pattern it is automatically selected and you may hit **Enter** to accept this class.

If the currently selected application includes the class it will be selected in the class pane. Otherwise you will be asked if you want to open a *ClassBrowser* on the class instead.

Find Application

Open a dialog box with a list of all applications containing the selected class. If you select an application in this dialog box, it will be selected in the application pane.

Settings...

Open a Class Settings dialog

Remove from Application

Remove the class from the application but not from the system.

Delete

Delete the class.

Methods Menu

The choices in this menu affect the currently selected method and provide automatic creation of access methods.

- New Method
- Create Get Method...
- Create Set method...
- Import...
- Execute
- Execute Inspect
- Find Application
- Senders
- Local Senders
- Implementors
- Local Implementors
- Messages w/o Implementors
- History
- Messages
- Remove from Application
- Delete

New Method

Create a new method. Display the default new method template for the selected application.

Create Get Method...

Create an access method for a specific variable. You will be asked to supply the names of one or more variables in a separate dialog box. The dialog box will contain only names of instance variables where the selector to be created does not conflict with an existing method. The generated code has the following form:

```
browsedApplications
        "Answer the value of browsedApplications."
    ^browsedApplications
```

Create Set method...

Create a method to set the value for a specific variable. You will be asked to supply the names of one or mores variable in a separate dialog box. The dialog box will contain only names of instance variables where the selector to be created does not conflict with an existing method. The generated code has the following form:

```
browsedApplications: aValue
        "Set the value of browsedApplications."
    browsedApplications := aValue
```

Import...

Add existing methods of the selected class to the application. You can select the methods in a dialog box which displays all methods of the selected class which do not belong to the selected application already.

Execute

If the selected method is a class method just execute it. Otherwise create an instance of the selected class with the message #new and execute the selected method with this object.

Execute Inspect

Same as Method/Execute but open a inspector on the result.

Find Application

Open a dialog box with a list of all application which containing the selected method. If you select an application in this dialog box, it will be selected in the application browser.

Senders

Open a method browser on the senders of the selected method.

If no method is selected prompt for a string. This string is used like selected text in the text pane. See Senders of selected text

Local Senders

Open a method browser on the senders of the selected method that are in subclasses of the selected class.

If no method is selected prompt for a string. This string is used like selected text in the text pane. See Senders of selected text

Implementors

Open a method browser on the implementors of the selected method.
Prompt for a method selector if no method is selected.

Local Implementors

Open a method browser on the implementors of the selected method that are in subclasses of the selected class.

Prompt for a method selector if no method is selected.

Messages w/o Implementors

Make a report with symbols occurring in the selected methods that don't have implementing methods. This choice is automatically activated if the *Automatic ST-Lint* is activated.

Use the Application/Settings/Config Variables choice to control which symbols are considered to be message sends.

History

Open a browser on all history versions of the selected method.

There is a methods list in the upper pane of this browser containing a list of old versions of methods and their date and time of their creation. This date is nil if the log entry is a system definition.

Messages

Open a message browser on the selected method.

Remove from Application

Remove the selected methods from application.

Delete

Remove the selected methods from the system.

ChapMan Application Browser Dialogs

The *ChapMan Application Browser* uses the following dialog windows.

Application Settings Dialog

This dialog contains various settings for an application. This dialog is also displayed if a new application is created. The following settings are available:

Name
The name of the application. This may be of any length and may include spaces, but may not be equal to the name of another application.

Filename
The file name of the application without extension. The application will be saved under this base name and the extensions **.CHA** and **.CLS**.

Organization Only Application
This indicates if the application is an organization only application.

Extra symbols to be ignored
A list of symbols that are not considered to be message sends by the Smalltalk Lint check.

Class Documentation Dialog

This dialog offers some options for the generation of class documentations. Under **Categories** you find a collection of all categories for all classes the documentation is be generated for. You may select the categories the methods of which you want to include in the output.

Under **Output format** you may select in which format the documentation should be generated. The **ASCII** choice creates a plain ASCII documentation. The width in characters for this format can be entered in the Global Settings Dialog. The two RTF choices create the documentation in the RTF format which can be printed by most popular word processors. Due to a bug in the AmiPro import filter, a special choice for AmiPro is available.

Class Settings Dialog

Under **Extra Symbols to be ignored** you can specify the symbols that should be ignored in the search for undefined message sends.

Keys Help

Apart from Digitalk's key assignments ChapMan offers the following accelerator key:

Ctrl-O
    Format Comment

Alt-L
    Enable or disable the *Automatic ST-Lint*

ChapMan Source Browser

ChapMan Source Browser

The *ChapMan Source Browser* reads an ordinary Smalltalk source file in chunk format and provides a means for manipulating and managing the changes contained in it. In addition it is used as an replacement for Digitalk's Method Browser.

See the Introduction for important general information.

See one of the following topics to get further information on the ChapMan Source Browser Window.

- Menu Description

Menu Description

The ChapMan Source Browser window offers the following menus:

- Menu: Change
- Menu: Display

Change

The following choices are available from the **Change** menu:

- Install
- Invert Mark
- Add to Application
- Remove
- History
- Senders
- Local Senders
- Implementors
- Local Implementors
- Messages
- All Displayed

Install

Install the selected change. The action depends on the type of the selected change:

Method definition
Install the method

Remove selector or class
Remove the specified selector or class

Evaluate
Evaluate the expression.

Define Class
Install the class in the system.

Saved Image, Open a window, Comment
Do nothing.

ChapMan Variable assignments, Class Comments
Set the variable or comment text.

Invert Mark

Invert the current marking state of the selected change, i.e., mark it if it is currently unmarked and vice versa.

Add to Application

Lets you select an application from a dialog box. If the selected   change is a method, it will be added to the application. If it is a class definition, it will be added to the application as an owned   class.

Remove

Remove the selected item from the displayed list. You will be asked if you want to remove the selected item   from the system also.

History

If the selected change is a method definition, open a browser on all history versions of the selected method definition. See Method/History for further information.

Senders

Open a *ChapMan Source Browser* on all senders of the selected method selector.

Local Senders

Open a *ChapMan Source Browser* with all senders of the selected method selector, which are in subclasses   of the selected class.

Implementors

Open a *ChapMan Source Browser* with all implementors of the selected method selector.

Local Implementors

Open a *ChapMan Source Browser* with all implementors of the selected method selector, which are in subclasses of the selected class.

Messages

Open a message browser on the selected method.

All Displayed

This is a submenu, the choices of which influence all displayed changes. The available choices are:

Install

Install all selected changes. See Install for more information how the different types of changes are processed.

Mark

Mark all displayed changes.

Invert Mark

Invert the marking state of all displayed changes, i.e., marked changes become unmarked and vice versa.

Add to Application

Let you select an application to which the displayed changes should be added. See Change/Add to Application to see how different change types are handled.

Remove

Remove all displayed item from the list of displayed items. You will be asked whether you want to remove all items from the system also.

Spawn

Open a new *ChapMan Source Browser* on the currently displayed changes.

Change Class Name

You are prompted to supply a name change expression in the form oldclassname>newclassname. This causes all method definitions in the displayed changes referring to the class with name oldclassname to be replaced with references to the name newclassname.

This is useful to resolve system conflicts when you are trying to install a class but there is already another class in the system with the same name.

Mark System Conflicts

Mark all the displayed changes that interfere with the system in some way.

Report Undefined Classes

Report all classes referred to in the displayed changes but that don't have a definition in the system or among the displayed changes.

Display

This menu provides various choices which let you determine, which messages should be displayed. There are four groups of choices.

1. In the first group you find

   - Class Filter
   - Saved Image Items
   - Enter Class Filter

   These choices cause only some of the change types to be displayed, and act as filters. An active choice has a checkmark in the menu. Only one of them can be active at a time.

2. The next three choices are filters also, but more than one of them may be active concurrently. These are

   - Later Items Only
   - Recent Versions
   - System Conflicts

3. The next three choices define filters which select the changes by their marking state:

   - Displayed Marked
   - Display Unmarked
   - Display All   There is exactly one of them active simultaneously.

4. The last choice is System definition.

See the specific section for more information.

Display Class Filter

If the selected change is a method definition, only change items with the same base class referring to the same base class as the selected change will be displayed.

Otherwise change items with the same type as the   selected change will be displayed.

Enter Class Filter

You are allowed to enter the following strings items:

Any string not containing '*'
This is considered to be a class name.

Any string containing '*'
This is considered to be a pattern which is matched against the string representation of the change item (as it appears in the list pane).

#comment
For pure comments.

#evaluate
For evaluate expressions

#defineClass
For class definitions

#method
For method definitions

#removeSelector
For selector removals

#removeClass
For class removals

#savedImage
For 'Saved Image' messages

#open
For open window expressions

#chapManVariable
For assignments to ChapMan variables

#classComment
For class comments

Saved Image Items

Display 'Saved Image' change items only. This is actually the   same as entering '#savedImage' in the Enter Class Filter menu choice.

Later Items Only

This choice is available only if a change item has been selected. Selecting this menu choice causes only changes occurring after the selected change to be displayed.

Recent Versions Only

Display the most recent version of each change item only, i.e.,   if the list of displayed changes includes two definitions for a method or class, display only the later one.

System Conflicts

Display only changes, which interfere with current definitions in the system.

Display Marked

Display only the marked changes.

Display Unmarked

Display only the unmarked changes.

Display All

Display both the marked and the unmarked changes.

System Definition

Toggle between the definition of the method in the current   system and the definition in the changes file.

ChapMan Library Builder

ChapMan Library Builder

The *ChapMan Library Builder* lets you define scripts which save an application to a Smalltalk Link Library (SLL) file.

See the Introduction for important general information.

Select one of the following choices to get further information on the ChapMan Source Browser Window.

- o Window Description
- o Menu Description
- o Library Builder Dialog

Window Description

The *ChapMan Library Builder* has two panes. The upper one is a list pane containing defined library builder scripts, and the lower one is a text pane containing selected script.

Menu Description

The ChapMan Library Builder window offers the following menus:

- Menu: Script

Script menu

The following choices are available from the **Script** menu:

New Script

A new script for the selected application will be generated. You will be asked to supply name for the script. You may then enter several options in the displayed Library Builder Dialog. ChapMan will then create a Smalltalk Library Builder script which will save the application into an SLL file. You may always adapt this script to your needs, in case that the default choices are not suitable or you have to do some special things during the library build.

Save & Execute

Save and execute the script in the text pane. After the execution of the script the ChapMan Library Builder tries to display two windows. The first is titled 'Library Imports' and displays all external references of the library. These classes, method or variables must be present at the time the library will be bound to the image. If you detect that some of these references should be in the library just add the object with one of the statements described below. The second window is titled 'Library Contents'. This window can however only be displayed, if the library is in a directory contained in the operating system's PATH-statement. It will contain all classes, methods and other objects added to the library either directly or implicitly. If you decide that one of these objects should not be contained in the library you should specify an import reference with the statements described below.

The following statements may be used in library builder scripts. See chapter 10 of you *Visual Smalltalk Programming Reference* for more information on the Smalltalk library builder.

sll addGlobalNamed: aSymbol

Add the named global to the library.

sll addPoolVariable: anObject named: variableName in: poolName

Add the pool variable <anObject> named <variableName> from the pool named <poolName> to the library being built.

sll addClass: aClass

This is the same as *sll addClass: aClass includeMethods: true*.

sll addClass: aClass includeMethods: includeMethods

Add <aClass> to the library. If <includeMethods> is true, then include all the class's methods. If <includeMethods> is false, then do not include any methods. If <includeMethods> is a collection of selectors, then only those methods will be included. If <includeMethods> is a Boolean and <aClass> is a class (not a metaclass), then the class's metaclass is added to the receiver too.

sll addMethod: aCompiledMethod

Add <aCompiledMethod> to the library. Include its source if the source is included for this library by default.

sll addMethod: aCompiledMethod includeSource: aBoolean

Add <aCompiledMethod> to the library. Include its source if <aBoolean> is true.

sll remap: anObject to: anotherObject

Change <anObject> to <anotherObject> in the library. You may use this if you want to <anotherObject> to by replaced for <anObject> when the library is loaded.

sll bindAction: anEvaluableAction

Set the action to be evaluated when the library is bound. For example

```
sll bindAction: [ChApplication initialize].
```

sll unbindAction: anEvaluableAction

Set the action to be evaluated when the library is unbound.

Execute All

Execute all scripts of the selected application. Errors are sent to the Transcript pane.

Remove

Remove the selected script from the selected application.

Library Builder Dialog

This dialog lets you select various options before the ChapMan creates a library builder script for you.

Filename
: The file name without any path information where the library should be saved to. You may use a question mark ('?') to indicate where the platform-specific letter for the library should be inserted. For example entering the file name 'Chap?20' will save the library to the file 'ChapO20.sll' on an OS/2 system and to the file 'ChapW20.sll' on a Windows system.

Directory
: Here you may specify a directory where the library should be created in. If you leave this field empty, the library will be build in the current directory.

Description
: Enter any text describing the library

Include Source
: If this box is checked, the source of methods will be included in the library.

Separate Source File
: If this box is checked. the source is saved to a separate file with extension SML rather then included in the SLL file.

Remap Class Variables to nil
: If this box is checked, all class variables of classes included in the library will be remapped to nil, i.e., they will be loaded as nil values when the library will be bound to the image. If the box is unchecked, the values currently stored in the class variables will be included in the library.

ChapMan IPF Browser

ChapMan IPF Browser

The *ChapMan IPF Browser* lets you create and browse *OS/2 IPF* for documentation and online help.

If you want to browse existing IPF files be sure that the main tags (i.e., **:userdoc.**, **:h**n **:euserdoc.** tags) start on a new line. Otherwise the browser will group more than one section into one heading.

See the Introduction for important general information.

Select one of the following choices to get further information on the ChapMan Source Browser Window.

- Window Description
- Menu Description

Window description

The *ChapMan IPF Browser* has two panes. The upper one is a list pane containing the headings of the IPF file, and the lower one is a text pane containing the text for the selected or new heading.

The title bar shows the name of the currently loaded IPF file or <untitled> if no file is loaded.

An asterix '*' in front of the file name indicates that the file been modified since the last save.

Menu Description

The ChapMan IPF Browser window offers the following menus:

- File
- Edit
- Heading

File Menu

This menu differs from other ChapMan windows in operations launched by the various choices (although the menu item names may be the same).

New
Clear the window to start editing a new IPF file.

Open
Open an existing IPF file.

Save
Save the current file.

Save As
Save the current file under a new name. It has the **Alt-S** short cut assigned. This means if you want to save the contents of the text pane only you must not press *Alt-S* because this will not only save the contents of the text pane but the file also. To the save only the contents of the text pane you must use the text popup menu.

Edit

This menu provides the familiar editing actions which all refer to the text pane.

Heading Menu

- Delete Heading
- New Subheading
- New Heading After Current
- Parse Menu Definitions...
- Collapse All
- Expand All
- Search...
- Search Next

Delete Heading

Delete the selected heading after a confirmation.

New Subheading

Add a new subheading to the selected heading.

New Heading After Current

Add a new heading as a sibling of the selected heading one position after the selected heading.

Parse Menu Definitions...

First open a dialog box where you can select the class which you want to be parsed for menu definitions. The selected class is then parsed for instance methods containing menu definitions, and you are again prompted to select the methods you want to be parsed. You may select several methods at the same time.

The IPF Browser then tries to extract the menu definitions from the selected methods in inserts the created headings after the selected heading in the heading list.

Collapse All

Collapse all headings.

Expand All

Expand all headings.

Search...

Search for a pattern which may contain wild card characters. If the text of a heading matches the provided pattern it is selected and the matched text is selected in the text pane. In addition the **Find Again** choice from the edit menu may be used to find the next occurrence in the text pane (this does not work for text containing wild card characters).

Search Next

Search for the next heading matching the pattern provided in the Search choice.

ChapMan Template Browser

ChapMan Template Browser

The *ChapMan Template Browser* lets you view and modify text variables and templates for a specify application or the global default values.

See Introduction for important general information.

See one of the following topics to get further information on the ChapMan Template Browser Window.

- o Window Description
- o Menu Description
- o Application Variables Description

Window description

The *ChapMan Template Browser* has two panes. The upper one is a list pane containing the application variables for the application displayed in the window title, and the lower one is a text pane containing the text for the selected variable.

Menu Description

The ChapMan Template browser has no special menus. Use the File/Save menu choice to accept the text in text pane for the selected application variable.

Application Variables

The application templates are divided into two sections. The first one consists of templates that are different for all applications. They are:

- Comments
- Init Code
- Finalize Code

The second section contains template variables that are likely to be the same for many applications. These variables may also be set to the string *<default>* which causes the default definition to be used. The variables in the second sections are:

- Class Template
- Method Template
- Method Comment Template
- Copyright

For a newly created application the variables in the first section variables are set to the default values, whereas the variable from the second section are set to the string *<default>*.

Text Variable: Comments

Enter a description or comments for the application here. This text will be displayed in the text pane any time you select this application.

Text Variable: Init Code

Enter Smalltalk code here which is evaluated before any of the class and method definition are filed in.

Text Variable: Finalize Code

Enter Smalltalk code here which is evaluated after all classes and methods have been filed in.

Text Variable: Class Template

Enter a class template here which will be automatically displayed when a new class is created in the application manager.

You can use macros in this template. See Template macros

Text Variable: Method Template

Enter a method template here which will be automatically displayed when a new method is created in the application manager.

You can use macros in this template. See Template macros

Text Variable: Method Comment Template

Enter a template which will be inserted into the source of a method each time this method is updated (not in new methods). If you save this template as an empty string, nothing will be inserted.

You can use macros in this template. See Template macros

For example

```
"@<date> <pid><:ask>"
```

You will be asked for a comment each time you save a method definition. At that time you have three options:

1. Enter short comment for the change you made to the method. This will be appended as a separated comment and marked with the programmer id (Pid) and current date.

2. Press **Enter**. This will only create a mark with the current date and the programmer id.

3. Press **Esc**. This will add no comment lines at all. This is useful if you just entered a comment and you don't want to have a second mark in the method.

Text Variable: Copyright

Enter a copyright notice here which can be expanded with the <copyright> macro in method or class template definitions. See Method Template for an example.

Template macros

The following macros are allowed:

<date>
    The current date.

<pid>
    The programmer identification. See Global Settings Dialog

<copyright>
    The copyright notice. See Copyright.

<:ask>
    Ask for string from the user and replace <:ask> with the supplied string. If the string is not empty, ': ' is inserted in front of the string.

All macros are case sensitive.

General Dialog Windows

Every ChapMan window except the *IPF Browser* has an options menu containing the following choices:

- Global Settings Dialog
- Symbols/Libraries Dialog
- **ChapMan: Auto ST-Lint**

The last choices is just a short cut for the corresponding setting in the Global Settings Dialog.

Global Settings Dialog

The global settings dialog contains the following entries:

Perform Auto-ST-Lint in the background
Default: true. When this choice is checked the Smalltalk Lint will be automatically started after each method save in ChapMan windows. This checkmark can also be toggled from the *Options* menu or with the **Ctrl-L** shortcut.

Exclude window events from Lint search
Default: true. The Smalltalk Lint does not consider names of window events to be message sends when this choice is checked.

Comment Width
Default: 50. Indicate the width for the 'Format Comment' command.

ReportWidth (Integer)
Default: 65. Indicate the width for various reports in ascii format.

Programmer Identification
Enter a programmer id here. This is used for automatic comment generation for methods and may be accessed by the **<Pid>** macro for templates.

Example with Pid = 'ch'

```
textVariables
        "Answer a collection of text variables."
        "@18.02.93 ch: FileName support added"
```

FileoutRequiredClasses (Boolean)
Default: false. If this variable is true, the definition of required classes will be included in the *.cls file, when an application is filed out.

FileoutRequiredClasses (Boolean)
Default: false. If this variable is true, the definition of required classes will be included in the *.cls file, when an application is filed out.

Application Directory
Default: Current directory. Application file names supplied in the Application Settings Dialog are considered relative to this directory.

Backup copies to keep of application files

Default: 3. Specify how many backup copies should be kept of application files (must be between 0 and 10). The extensions are **CH0** to **CH9** for **CHA** files and **CL0** to **CL9** for **CLS**-files., where the files names containing the 0 are the most recent copies.

Confirm class and method deletions

Default: Checked. If this box is checked all class and method deletions are subject to a prior confirmation.

Default Method Category Format

In this group box you can select one of the available method category formats. See Method Categories for more information.

Symbols/Libraries Dialog

In this dialog you can define which SLLs are considered to be system SLLs and which symbols should be ignored in the search for undefined message sends.

Extra Symbols to be ignored
Specify the messages which should be ignored in the search for undefined message sends.

User/System Libraries
Define which SLLs are considered to be system SLLs or user SLLs. You may toggle the state with the **User** and **System** buttons.

Product Information

Select this choice to get information in the product you are using.