Preface

The aim of this textbook

The goal of this textbook is to introduce concepts of object-oriented software development and the programming language Smalltalk as a vehicle for their implementation.

Why object-oriented software development

After the introduction and experience with a series of programming paradigms, object-oriented (OO) software development has become the undisputed mainstay approach to the development of modern software. The reason for this is that the OO paradigm best satisfies demands for fast development of reliable and maintainable software. Its principle of viewing software as modules modeling real world objects is a natural reflection of both clients' and developers' view of the world, and provides a basis for building extendible and reusable libraries of individual objects and whole frameworks that can be relatively efficiently extended or customized to build new applications.

Why Smalltalk

Besides being the first full-fledged commercial object-oriented programming language, Smalltalk is one of the most exciting and powerful programming languages. The fact that most programmers who learn it prefer never to use any other language is a testimony to this. Its uncompromising reliance on OO principles and its basic simplicity also mean that if you want to learn object-oriented programming, there is nothing in Smalltalk to distract you. In Smalltalk you cannot do anything without objects and the current shift towards object-oriented programming thus makes Smalltalk the perfect vehicle for the development of a sound OO perspective. Very few languages offer such an undiluted environment. Smalltalk implementations are also very interactive and this makes Smalltalk an ideal tool for experimentation and hands-on classroom presentations.

Another interesting feature of Smalltalk is that it is not only a language but also a complete programming environment with tools such as browsers, object inspectors, debuggers, and file editors. While this is not a rarity any more, Smalltalk still distinguishes itself by the fact that it contains all the source code of its environment and its own implementation, allowing the user to study the principles of sophisticated OO design, and to make personal modifications to the environment if desired.

Since the implementation of a program development environment requires most of the building modules that an application requires, all the essential building blocks already in the environment's library. Since libraries of other OO language have been largely derived from Smalltalk, understanding the Smalltalk library is also an excellent introduction to libraries of other OO languages as well. The simplicity of Smalltalk makes the study of its architecture much easier than it ever could be in most other languages.

Because the environment contains its source code, you can learn much about Smalltalk directly from its implementation. Since the source code includes the compiler, the programming tools, user interfaces, and operating system components, Smalltalk is also excellent for learning about most software-related issues of Computer Science. This includes algorithms and data structures, data base management systems (both relational and object-oriented), operating systems, compilers, networks, graphics, simulation, and others. Whole Computer Science curricula can be based on Smalltalk.

Those with a pragmatic outlook will appreciate that Smalltalk is currently, after C++, the second most popular object-oriented language for large software projects, both stand-alone and network-based. It is also the second fastest growing object-oriented programming language after Java. As a consequence, there is a considerable shortage of qualified Smalltalk programmers and mastering the language is a very good investment.

Why has Smalltalk been so popular in commercial applications? The main advantages of Smalltalk over other languages are its conceptual simplicity and its undisputed superiority for prototyping and rapid application development. The high productivity of Smalltalk development is due to the ease with which new

Smalltalk code be implemented and tested which is directly related to its design principles, and to the very large library of reusable and easily extendible built-in parts. Smalltalk's excellent support for user interface development is another major reason for its popularity.

The most popular Smalltalk implementations are also very portable among many hardware and operating system platforms. As an example, the VisualWorks Smalltalk applications developed in this book will run without any changes on 13 different platforms including Microsoft Windows, OS/2, the Macintosh and the SUN. It is even possible to develop different parts of a non-trivial Smalltalk application on different platforms and merge them together. This high degree of portability is a great economical advantage not available in most other languages.

An important consequence of the fact that Smalltalk is so popular is that its developers must keep up to date with the latest technological advances. As a result, Smalltalk was, for example, one of the first languages to provide advanced extensions to distributed objects. It is also the language in which some of the most powerful and popular development environments for other languages such as VisualAge Java and C++ have been developed.

Finally, developers appreciate Smalltalk because it is very mature - it has been in existence for almost 30, and in productive use for almost 20 years. This is almost unbelievable when you consider that even after all these years it remains one of the most progressive programming environments available. The OO principles of all major OO languages in use today have been built on the basis of the Smalltalk experience and are still borrowing principles from it, without ever quite achieving its power and simple elegance, in our opinion.

How this book differs from other Smalltalk books

Although there are many excellent texts about Smalltalk, this is the only one that covers all the following topics

- OO concepts
- OO application development
- Smalltalk programming environment including its user interface painting tools
- Smalltalk base classes
- elements of Smalltalk style

illustrates them on many examples, both small and large, and provides numerous exercises and projects that challenge the reader to master the material by personal experimentation. You can download the software presented in the text from the author's home page at http://www.acadiau.ca/ivan.

The book does not assume any previous programming background but the subject matter is not limited to essentials and provides enough depth to allow the reader to write substantial applications and to understand the deeper principles of Smalltalk. There is, however, no way that one could write a book of this size and introduce cover all functionality of Smalltalk. We thus acknowledge that in spite of the breadth of our coverage, many essential topics that are not covered in great depth (such as exceptions or more complicated application architectures) or that remain uncovered. As an example, we have not touched the interface between Smalltalk and databases, the use of Smalltalk for network programming, interfacing Smalltalk to external applications, and other subjects. We also have not covered all graphical user interface widgets, code stripping, team development tools, and other important but more advanced subjects.

Besides the unparalleled completeness of coverage, the structure of this book is also unusual. We begin with a presentation of the principles of the object-oriented paradigm, continue with a rather detailed introduction to object-oriented analysis and design, and then present both the essential components of the Smalltalk library and its user interface building tools and components. User interfaces are introduced rather early and in relative detail because we think that in this age of sophisticated computer applications one should build programs that not only do interesting things but also provide a pleasing user interface. Besides, creating programs with modern user interfaces increases the enjoyment of programming and makes its study more stimulating. And Smalltalk makes this rather easy.

The scope of this book and its intended audience

Since the book does not make any assumptions about the reader's background, it is suitable for newcomers to Smalltalk and even those who never studied programming. In fact, we use the text in a first year programming course for Computer Science students. However, since the scope goes well beyond essentials the book is also of interest to those who already know another programming language and want to learn object-oriented programming or Smalltalk.

Readers who already know Smalltalk will also find the book useful because the presentation is quite different from that of other books and includes material not found elsewhere. This includes examples of implementation of conventional abstract data types, principles of implementation of customized user interfaces, processes, and a discussion of metaclasses with examples of metaprogramming. Besides, Smalltalk is always worth looking at from a different perspective because there is never an end to learning something new about this rich and flexible programming environment.

How to use this book

.

Whether you are using this book in a course or for self-study, we recommend the following approach: Study the material, try the examples in the text, and do as many exercises as possible to become comfortable with Smalltalk and its environment and program development skills. As you progress, implement some of the projects listed in the Appendix 6 using the approach presented in the main text.

Which Smalltalk

Smalltalk is currently undergoing standardization and many of the classes in its library are now required in all compliant products. However, there is no way to prescribe all details of an inherently extensible environment such as Smalltalk. As a consequence, individual Smalltalk dialects will always differ from one another in implementation, design, user interfaces, and even conceptual models of its building blocks. From the point of view of a Smalltalk programmer, especially a novice, the main differences between the dialects are in the user interface of the programming environment and the number and implementation of the built-in user interface components. Support for graphics, the structure of windows and their components, and support for file operations also differ from one dialect to another.

Because of the differences between different implementations, every Smalltalk book that goes beyond basics must select one particular dialect and cover language-dependent features as implemented in this dialect. We chose VisualWorks because this happens to be the dialect that we have used for the longest time and because it is the dialect most directly related to original Smalltalk. If you don’t use VisualWorks Smalltalk, your user interface will be somewhat different from the illustrations in this book, some tools such as user interface painters will be different, and the corresponding classes will be implemented or organized differently. However, most of the material presented in this book applies to any Smalltalk dialect without any change. See Appendix 8 for a listing of commercial and free Smalltalk products available at the time of this writing.

Acknowledgments

Like many authors who wrote about Smalltalk before me, I want to express my thanks and admiration to the group of visionaries who invented Smalltalk and to those who keep extending it creatively and applying it to new domains.

I am very grateful to the many readers of the preliminary forms of the manuscript who taught me a lot by their criticism and suggested ideas that are now implemented in the text. I am also obliged to my students whose blank looks alerted me to inappropriate presentation strategies and whose improvements of classroom examples served to improve the text. I hope that I will get similar constructive suggestions for improvements from the readers of this book as well. Finally, I must acknowledge the influence of the Smalltalk newsgroup on Internet at comp.lang.smalltalk. I learned a lot from it and borrowed some of the ideas expressed at this forum for examples and exercises.

I have been teaching Smalltalk for the last five years and forgotten many of the students who actively shaped my understanding of Smalltalk and influenced my teaching, but I wish to thank the following students and friends whose names stand out in my memory. They include (alphabetically) Fauzi

Ali, James Benedict, Peter Burka, Randy Giffen, Kenn Hussey, James Moody, Oliver Oey, Mark Rhodenizer, Ravi Palepu, and Donald Smith. Most of them are now employed as professional Smalltalk programmers.

Finally, but most importantly, I wish to thank my family. During the long time that it took to write this book, my wife Jana managed to pretend that spending the time that other people devote to holidays on my computer was a part of normal life, smoothed over my continuous swings between fascination and frustration with this book, and added a human dimension to my life. Our three children - Ivan, Ondrej, and Dominika – helped greatly to maintain this illusion. I have been very lucky to have a family that provided such a warm, supportive, and stimulating environment.