9 Towards An Integrated Environment: A Prototype for Harmony

"Build me straight, O worthy Master!
Staunch and strong, a goodly vessel,
That shall laugh at all disaster,
And with wave and whirlwind wrestle!"

[H. W. Longfellow, The Building of the Ship, 1849]

9.1 Introduction

In this chapter the overall design of the specification environment entitled Harmony is presented and details are given regarding both its operational capabilities and its GUI appearance. This tool is intended to fully support the modelling process proposed in Chapter 7, with special attention paid to provisions for sustaining the formalisation of UML classes and state diagrams described in Chapter 6. Deformalisation is also supported and aids are included for easy manipulation of Z++ symbols in the process of writing formal specifications. A "tandem" mode of operation is introduced, consisting essentially in the synchronised presentation of a UML class compound and its corresponding Z++ class specification. A more complete description of this CASE tool, currently evolving into a prototype, is provided in Appendix C. In this chapter, the general principles of Harmony are presented first, followed by an overall view of the environment, and then by successive descriptions of Harmony's main components: the Project Pane, the UML Space, and the Z++ Space. A number of additional features of the environment, including specific toolboxes and buttons, are also presented.

9.2 General Principles

Because we attempt to combine the benefits of "both worlds" ("formal" and, respectively, "informal," as well put by Alexander in the title of his paper [Alexander95]) and to "harmonise" the use of the UML and Z++ notations we have assigned the name Harmony to the environment that supports our modelling approach. In its present form, Harmony is intended to fully support the modelling process outlined in Chapter 7, with particular emphasis on the formalisation and deformalisation activities described in Chapter 6. Both UML and Z++ specifications can be "simultaneously" developed in Harmony and a bi-directional link exists between the formal and the graphical representations of the system, ensured by the translation mechanisms described previously in the thesis. This allows changes in the graphical representations to be reflected into the formal specifications, as well as the modifications of the formal part to be fed back into the diagrammatic description of the system.

Reflecting our philosophy for a rigorous yet pragmatic modelling approach, our goal for this stage of Harmony's development was to keep things simple and focus on those aspects of TCS that must be completely and correctly captured during the early stages of software development. Further extensions for the environment are possible, some of them outlined in the Conclusions of this thesis, and provisions for interfacing with external tools are included. Presently, Harmony's designed capabilities are adequate for the development of the Elevator case study described in Chapter 8, as well as for modelling other TCS. Of course, software systems in which the focus is not on temporal restrictions ("non TCS") can also be specified using Harmony.

The environment operates on specification projects, which are sets of specifications represented in diagrammatical (UML) and/or mathematical (Z++) forms. For this reason Harmony is referred to as an integrated specification environment (ISE). The combined result of the specification activities supported by Harmony, more exactly the sum of the artefacts (model elements) produced in these activities within the procedural frame presented

in Chapter 7, constitutes the integrated model of the system. Since a total formalisation of the system is typically not required, the environment can be used for a partial formalisation within a complete specification of the system. In practical terms, this means that it is not necessary that a one-to-one correspondence between UML and Z++ components is achieved –some parts of the system will be described both in UML and Z++, while other only in UML or only in Z++. In principle, it is possible to have the entire system specified solely in Z++, but this can be efficient only in the case of small-sized systems (it also comes against our idea of combining formality and informality in software specification).

One of the distinguishing characteristics of Harmony is that it is monolithic in the sense defined in Section 4.2. By itself, it is sufficient to sustain a complete specification of the system and through its provisions for interfacing with external tools it is also capable in principle of supporting further development (in particular, we envisage interfacing with external tools for formal proof and formal refinement). Thus, it can be used as the "root" instrument from which other applications can be started –this is in contrast with other approaches, for instance RoZeLink's, where three applications need be separately started from the operating system in order to perform the formalisation and deformalisation processes [RoZeLink99]. As shown in next section, it also "monolithically" integrates two "worlds," the visual world of UML, and the textual, intensively symbolical world of Z++.

Another point that needs be further highlighted about Harmony is that it is designed for producing pragmatic, efficient, and precise models through the combined used of semi-formal diagrams and formal specifications. In this respect, the use of Z++ has primarily the role of supporting better understanding of the system and, generally speaking, of enhancing the intellectual control over the system. As discussed in Chapter 2, this is one of the main advantages of using formal methods and, consequently, although more intricate features for formal processing can be added later through Harmony's "add-ins" feature, they nevertheless are beyond the scope of this thesis. Only a syntax and consistency checker is envisaged to be included directly in Harmony, this being a useful tool for enhancing the developer's confidence in the accuracy of his or her Z++ specifications.

As a matter of overall organisation of the two modelling spaces it is necessary to mention that while a project may consist of several class diagrams there is a unique Z++ specification for the system. Because a Z++ specification encompasses not only classes but also statements external to classes (such as definitions of global types and operations on classes) and because we have attempted to ensure a consistent way of accessing the groups of artefacts within the project, a Global Spec component has been included in the Z++ Space. When fully expanded, as detailed in Section 9.6, it shows the entire text of the formal specification.

On a more detailed level, support for the class compound construct introduced in Chapter 7 is available, the idea being that the key classes of TCS need be detailed not only in respect with their attributes and their operations, but also with the sequencing of their operations (as captured in state diagrams). From a GUI point of view, a simple splitter bar between the space in which the UML class is represented and the one that corresponds to the class' state diagram is introduced, the two regular UML constructs being syntactically associated in the graphical model (they are also inherently associated in Z++ class declarations, where, within the CLASS construct, the HISTORY clause describes the temporal properties of the class' objects).

Also, since an intense work on UML class compounds and on their corresponding Z++ classes is expected, a synchronisation mechanism of on-screen presentation of the corresponding COMP and ZPPC constructs is proposed (the abbreviations introduced in Table 7.I are used again in this chapter). This mechanism defines a mode of operation that can be viewed as a manifestation, in our terminology, of the tandem principle (simply put, it means that two entities are working together for accomplishing a common goal). This mode of operation that allows in essence the "simultaneous" development (or the simple inspection) of a class in both its UML and Z++ forms is further described in the next section.

Before describing further the organisation of Harmony, we need to point out that only the design of the environment has been completed, but not its implementation. Therefore, the screen shots that follow are only aids for further development and do not represent actual

captures of the running environment. The examples of UML and Z++ model elements included in Fig. 9.1, showing the Elevator class compound and its corresponding Z++ class specification have been pasted into the environment panes, and have not been developed with Harmony. The design of Harmony's user-interface presented in this chapter and further described in Appendix C, a "mock-up" prototype written in Java, includes nevertheless the necessary details for fully sustaining Harmony's implementation.

9.3 Overall Organisation

Fig. 9.1 presents Harmony in a typical situation, in which a project is loaded and work is undergoing in the UML and Z++ spaces on several modelling elements, specifically a use case, a scenario, a sequence diagrams, two UML class compounds, and two Z++ classes. As seen in the figure, the environment consists of a main window (or browser), divided into several panes and containing other GUI elements such as a menu bar and toolbars. Since a systematic description of Harmony's user interface is included in Appendix C, we focus in this chapter only on those aspects that distinguish the most this ISE. From this perspective, it is notable that Harmony has three main panes, referred to, respectively, as the Project Pane, The UML Space and the Z++ Space. In addition to these panes, to the menu bar, and to the environment's toolbars, a message console and a status bar are also included.

In short, the entire organisation of the project can be viewed in the Project Pane, which shows the collections of artefacts grouped as indicated in Chapter 7. Work on the semi-formal model is performed in the UML Space, and formal specifications are written in the Z++ Space. It is important to note that in this organisation the three panes can all coexist on the screen at any given moment, but they can also be individually turned off, as shown in the View Menu presented in Fig. 9.2. Thus, work can proceed either in parallel in the semi-formal and the informal spaces, or can be focused on one of the to modelling "worlds". The specifier can turn off either "world" and use only half of Harmony's capabilities, either for

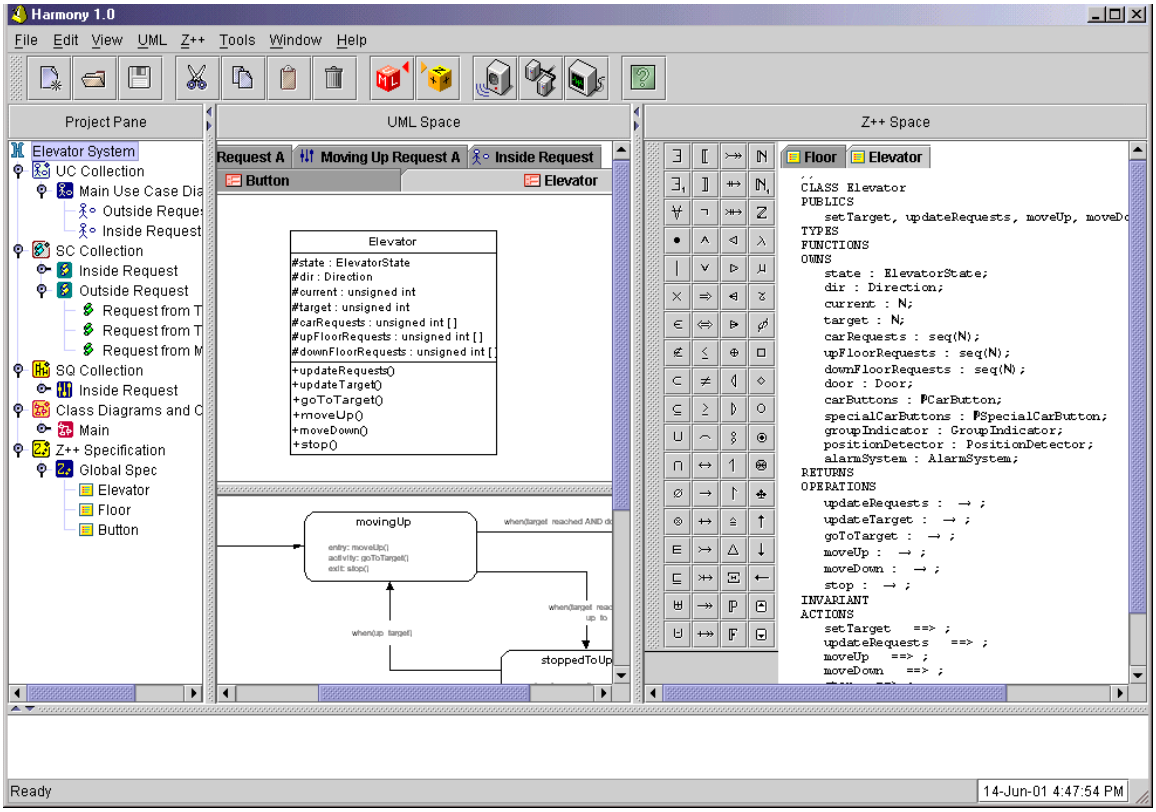

Fig. 9.1 Harmony's Look

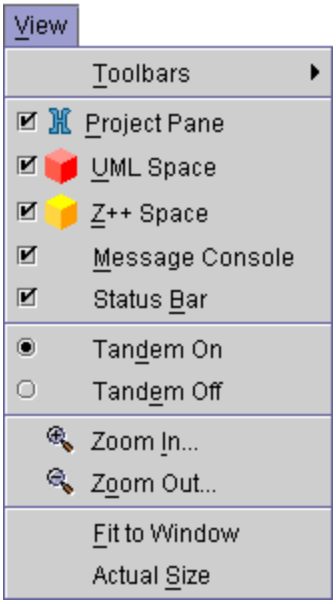

Fig. 9.2 The View Menu

developing solely UML models or for exclusively representing the system formally. As indicated in Fig. 9.2, all the panes of Harmony can be shown or hidden, although it will be of little value to have both the UML Space and the Z++ Space turned off simultaneously.

The same figure also allows the further description of the tandem mode of operation. In short, this mode of operation brings to the front of both modelling spaces (UML and Z++) the pair of corresponding COMP and ZPPC descriptions, irrespective on which space the developer is actually working. As such, all the relevant information about a class, specifically its UML structure in CLS, the UML state diagram CLSTD, and the formal ZPPC representation are visible at the same time on the screen provided that the "tandem option" is turned on and, of course, both UML and Z++ panes are open. The tandem mode of operations extends to dass diagrams and their counterpart, the entire Z++ specification as reflected by Global Spec.

9.4 The Project Pane

The Project Pane, shown in Fig. 9.3 with the ELS project loaded and partially completed, is one of the three principal areas of the Harmony window. Its role is to visually present the project's structure in terms of artefacts and groups of artefacts as described in Chapter 7 and to support a number of operations that allow the gradual development and organisation of the project. These operations consist of creating a new artefact or group of artefacts, moving an element from a group to another, and deleting an artefact or a group of artefacts. They are invoked by mouse actions within the pane's area, for instance a right-mouse click on the empty space of the pane opens the New Model Element Selector shown in Fig. 9.4. Two of the above operations are also available through other interface elements of Harmony, more precisely New is included in the File, UML, and Z++ Menus, and both New and Delete have icons on the environment's main toolbar (again, we refer to Appendix C for further details).

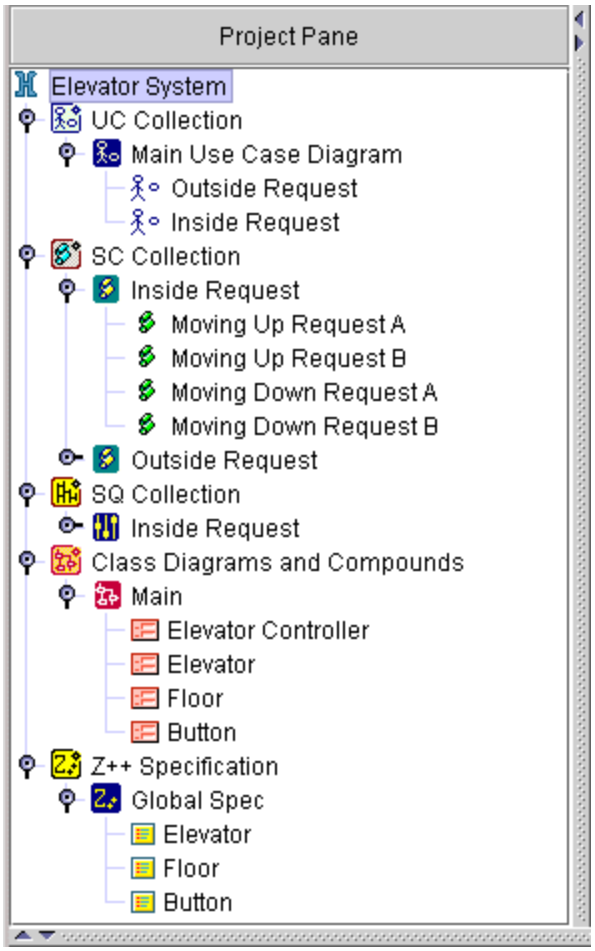

Fig. 9.3 The Project Pane

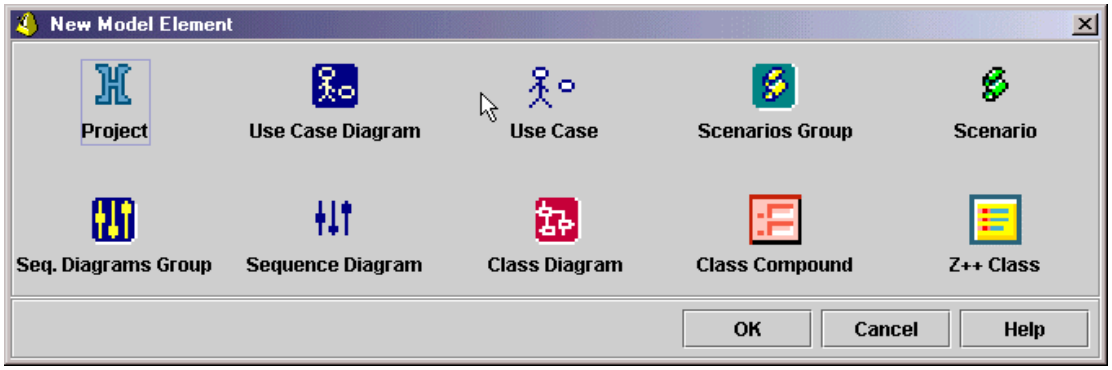

Fig. 9.4 The New Model Element Selector

From an operational point of view, immediately after Harmony is started all the environment's panes are empty, including the Project Pane. If from this state a new project is created, the view of Harmony is the one indicated in Fig. 9.5, which highlights the initial structure attributed by default to any project. This structure, following the guidelines of the procedural frame presented in Chapter 7, consists of the five major groups of artefacts considered there: the UC Collection, the SC Collection, the SQD Collection, the Class Diagrams and Compounds Section, and the Z++ Specification. (We prefer not to use the term collection for UML diagrams and compounds, since it may hint to an unstructured type of organisation, which is acceptable in the case of use cases, scenarios, and sequence diagrams –which need not, and typically cannot be completely specified,– but is not well suited for classes, which must be fully and correctly defined and organised).

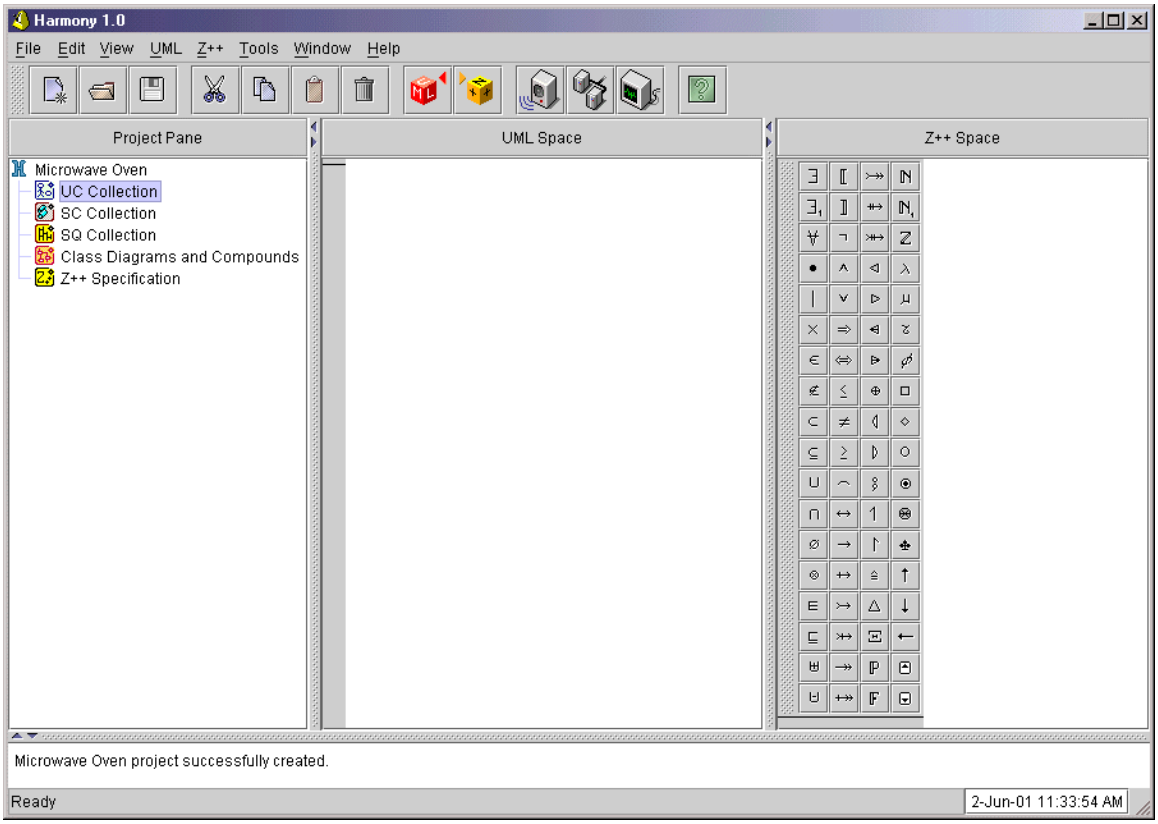

Fig. 9.5 Harmony with New Project Just Created

9.5 The UML Space

In the UML Space the specifier can work on one or more model elements, each model having its own tabbed-pane within this space. In Fig. 9.1, it can be seen that several such elements are opened at the same time, the one active being the Floor Class compound, with both its class specification and state diagram shown. Various options for working on the UML Space are available through the environment's menus and its toolbar. Among other things, the UML menu shown in Fig. 9.6 indicates that it is possible to disable the current UML toolbox (e.g., for inspection purposes, its elimination resulting into an increase of UML Space's visible area), as well as the state diagram part of a class compound (there are classes that have a trivial state diagram).

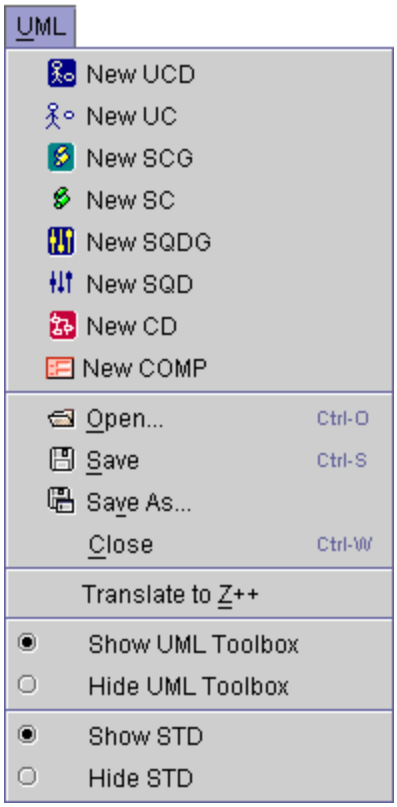

Fig. 9.6 The UML Menu

The same figure shows that creating new model elements, opening existing ones, saving them, or closing them are operations also possible through the UML Menu. An important function accessible via this menu is “Translate to Z++,” which allows the user to propagate new UML specifications or changes to existing ones into the Z++ Space. The rules for automated formalisation described in Chapter 6 are applied in this process.

The UML Space also contains a toolbox specific to each type of artefact created using the modelling approach proposed in this thesis. Since there are five distinct types of such artefacts, five types of UML toolboxes are provided, the one visible at a given moment corresponding to the type of artefact currently shown in the front tabbed-pane of the UML Space. One of these toolboxes is presented in Fig. 9.7, and all five are included in Appendix D. Some general symbols, such as “select item,” “text,” and “annotation” (the first three on the left-hand side of Fig. 9.7) are a common presence in most if not all UML toolboxes. The SQD Toolbox presented below includes additionally the “state,” “activation bar,” “message,” “message to self,” “asynchronous message,” “return message,” and “destroy object” symbols.

Fig. 9.7 A UML Toolbox

9.6 The Z++ Space

In Harmony, the Z++ space is the equal partner of the UML space and as such its dedicated menu has an organisation similar to that of the UML menu, as shown in Fig. 9.8. For instance, the reverse process of formalisation, the transfer of information from Z++ to UML, with its inherent simplifications discussed in Chapter 6, is invoked via the “Translate to UML” option, which has the counterpart “Translate to Z++” in the UML Menu.

Additionally, there are several functions specific to the Z++ space, all available through the Z++ menu.

Firstly, there is the “Analyse” option, which is intended to allow the syntax and consistency checking of the formal specifications. Secondly, due to the specific organisation of the Z++ specification, options for the presentation of the Global Spec as well of the Z++ classes are provided. More precisely, the Global Spec can show only the Z++ contents extraneous to classes, such as names of global types and hiding operations on classes (in case option Classes Hidden is selected); this contents together with the names of Z++ classes (if option Classes Collapsed is chosen); or the full text of the Z++ specification, i.e. the Z++ statements extraneous to classes and the detailed description of classes (if option Classes Expanded is selected). Also, an individual Z++ class can be presented within the UML space either alone (Hide Context option selected) or accompanied by the Z++ contents not included in classes (Show Context option selected).

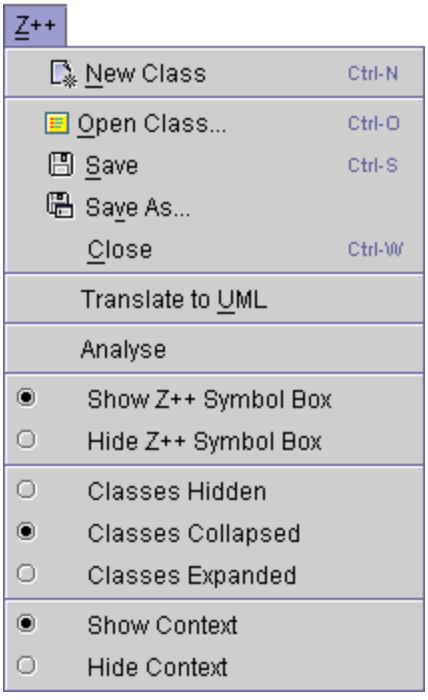

Fig. 9.8 Harmony’s Z++ Menu

A further distinction from the UML counterpart comes from the fact that a Symbol Box instead of a Toolbox is available when working in this space. This "palette of mathematical symbols" provides a practical alternative to the use of combinations of keystrokes for inserting special symbols in the formal specification (a similar Symbol Pallette is available in Logica's Z Formaliser [Formaliser01]). The Symbol Box for the Z++ Space, with its comprehensive set of items is presented in Fig. 9.9. The Z and Z++ specific symbols have been compiled from the indexes available in [Spivey92, pp. 153] and [Lano95, pp. 417-418] and the Symbol Box includes only those items that cannot be written using a standard font such as Courier or Times New Roman. For instance, in order to keep the Z++ Symbol Box as small as possible the arithmetic operators +, -, *, and / are not included, nor are Z specific notational elements such as ::=, < .. >, or >> that can be represented using regular fonts. Because in Z++ there is a need for subscripts (and sometimes for superscripts) two non-Z symbols, the superscript and the subscript indicators are also included as the last two elements of the Z++ Symbol Box. As for the organisation of this toolbar, a "topic related" criterion has been applied, the symbols being grouped according to their use: existential and universal quantifiers first, followed by statement separators, then by operators pertaining to sets and bags, then by function related symbols, etc. The nine elements in the Symbols Box that precede the superscript and the subscript indicators on the last row are Z++ specific (do not pertain to the regular Z). Further details on the Z++ Symbol Box are available in Appendix C.

Fig. 9.9 The Z++ Symbol Box

9.7 Other Features

There are other features available in the Harmony environment, which is nevertheless kept as simple as possible without jeopardising either its ease of use or its full support for the formalisation activities described in Chapter 6 and for the combined UML/Z++ modelling process proposed in Chapter 7. Some of these features are briefly described below, while additional details are available in Appendix C.

For instance, there are five environment-specific buttons visible on Harmony's main toolbar, namely the Translate to Z++, Translate to UML, Tandem Off, Tandem On, and Analyse buttons. In addition, the logo used for Harmony (taken from [RogersGifs01] Clipart Gallery), can also be considered environment specific. In order to exploit a bit the harmony metaphor, the first four symbols presented in Fig. 9.10 have icons related to the acoustic domain, specifically a metronome for the Harmony logo, a single (mono) audio-speaker for the Tandem Off button, a pair of speakers (a stereo system) for Tandem On, and a sound analyser for the Analyse Z++ Specifications function. The last three icons have been downloaded from [LeosIcons01] while all the other icons present in Harmony have been either taken from the "Java Look and Feel Graphics Repository" (standard symbols such as New, Open, Delete, etc.) [JavaLook01] or created by us from the scratch (all the symbols for the artefacts and all the elements of the UML and Z++ toolboxes). For translation operations between the "worlds" of UML and Z++ two new symbols have been designed, both using "transfer arrows" and cubes in their representation, the latter suggesting complex, well defined "worlds" (of modelling, in our case). These translation buttons are represented on the last two positions of Fig. 9.10.

Fig. 9.10 Harmony Specific Symbols

Regarding the Harmony logo it is interesting to note that it can be viewed as conveying a combination of suggestions about the two most distinguishing characteristics of our modelling approach: focused on smooth integration of notations (“harmony”, suggested by an instrument associated with the rhythm of music), and focused on temporal properties of systems (“the metronome,” a device which punctuates the passage of time).

Other icon-centred elements of Harmony’s user interface include a Legend Pane for the symbols used in the modelling process, accessible via the Help menu. One of the tabbed-pane of the Legend Pane is shown Fig. 9.11.

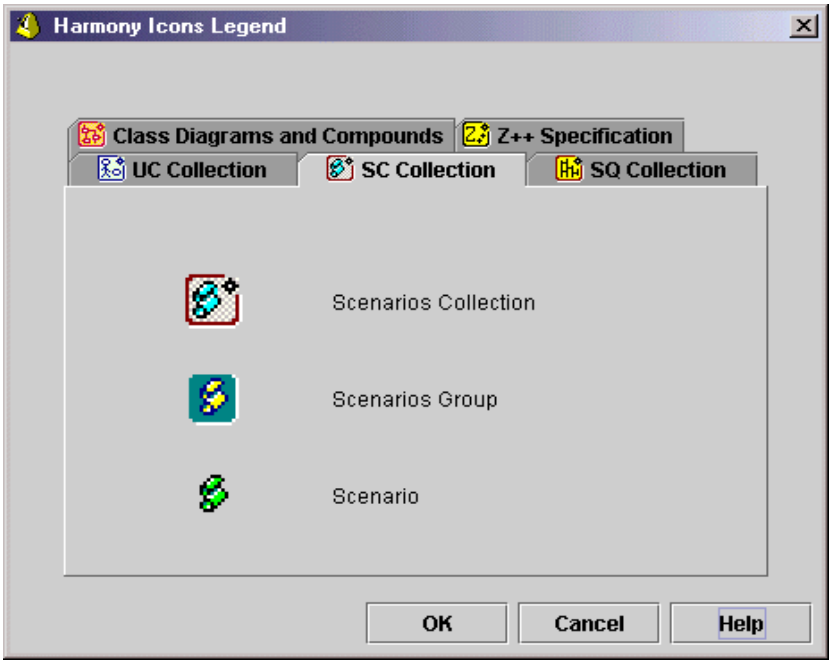

Fig. 9.11 The Legend Pane

Beside these detailed aspects of Harmony’s design there are some other, higher-level functional features that deserve to be mentioned. As shown in Appendix C, they include tools for customising the properties of the editor, of the project, or of the environment as whole, provisions for “add-ins,” zoom-in and zoom-out features, options for the export and the import of Z++ specifications, and creation of additional Harmony windows. In

particular, through the Add-Ins feature present in the Tools Menu, connections with external software tools are envisaged, and through the Export Z++ Specification included in the File Menu an independent file containing solely the Z++ description of the system can be generated for the purpose of being used in a separate development context. The counterpart of the latter feature, the Import Z++ Specification, has the role of allowing the inclusion of Z++ specifications developed externally into a Harmony project. This capability would permit the subsequent generation of the corresponding UML class structure through deformalisation.

9.8 Chapter Summary

In this Chapter the Harmony integrated specification environment has been introduced through the description of its user-interface and of the functionality available through this interface. This environment is intended to provide a monolithic integration of UML and Z++ notations by fully supporting the formalisation and deformalisation activities presented in Chapter 6 as well as the modelling process of TCS proposed in Chapter 7. The principles that permeate Harmony's design, the environment's general organisation, as well as its three major components, the Project Pane, the UML Space, and the Z++ Space have been described in a fair level of detail. Remarks on some secondary aspects of Harmony, such as specific icons and symbols, have also been included. Although Harmony is only in the design stage, the description presented in this chapter provides a good foundation for its implementation and allows the consideration of possible enhancements, some of them outlined in the next chapter.