10 Conclusions

"What we call the beginning is often the end.
And to make an end is to make a beginning.
The end is where we start from."

[T.S. Elliot, Little Gidding, Four Quartets, 1942]

10.1 Introduction

At the conclusion of this thesis, we first look back and highlight the merits and the limitations of our approach and then look forward to point out the work that remains to be done. To evaluate our work in the context of current research, a summary comparison with the closely related approaches discussed in Chapter 4 is included. The contributions of our work are presented by dividing them in two categories, principal and secondary, and the limitations of the present work are briefly reviewed. Needed improvements to the work described here and connected research paths that can be beneficially pursued in the future finalise the chapter.

10.2 Summary Comparison with Closely Related Approaches

In Chapter 4 five research studies were classified as "closely related approaches" to our work. Although they were examined in some detail in that chapter, we resort again to them in order to provide a brief comparison with our work. This comparison is based on a number of criteria, specifically:

- Type of translation from diagrammatic notations to formal specifications, which can be either an OO to an OO or an OO to non-OO formalisation (the latter means that

constructs of a non object-oriented formal language such as Z are adapted to represent OO constructs from the semi-formal counterpart);

- Provisions for modelling RT systems (either included or not);
- Type of integration of notations, based on the classification introduced in Section 4.2. Under this criterion, we denote simple formalisation (or derivation) by F, complementary formalisation by CF, and tight integration of notations (which involves two-way translations) by TF;
- The characteristic that can be referred to as the monolithic construction of the supporting specification environment (the definition of a monolithic environment has also been introduced in Section 4.2);
- Capability of applying tool-supported processing techniques on the formal specifications, including syntax and consistency checking, formal verification, and refinement. This capability describes the present situation and refers to the connection with tools that already exist;
- The usage of the formal notation involved, reflecting its popularity and the number of applications in which it has been employed.

The results of this comparison are shown in Table 10.I. From this Table it can be seen that our approach has its merits as well as its limitations. While the merits are stressed in Sections 10.3 and 10.4, about the two main limitations highlighted in the table we need to point out that although one is more difficult to overcome (specifically, it is difficult to match the popularity enjoyed by Z, ZEST, or Object-Z), the other (connection to tools for further formal processing) can be surmounted through the continuation of the work presented in this thesis (Section 10.6 describes our intentions in this respect). In addition, there are several other limitations, discussed in Section 10.5, which also can be overcome through additional work. Nevertheless, we believe that our approach provides a viable alternative for combined, semi-formal/formal software specification, and introduces a fresher presence in the landscape of pragmatic development of TCS through synergetic use of semi-formal and formal techniques.

Table 10.I Summary Comparison with Closely Related Approaches

Approach	Criteria					
	OO to OO formalisa-tion	RT specification capability	Type of integration of notations	Monolithic specification environment	Processing of formal specifications (analysis, refinement, etc.)	Usage of formal notation
[Jia97]	no	No	CF	no	Yes	high (Z)
[Noe00]	no	No	CF	partial	Yes	high (Z)
[France97]	no	Yes	F	no	Yes	high (Z)
[RoZeLink99]	yes	No	TF	no	Yes	high (ZEST)
[Kim00b]	yes	yes	CF	N/A	Yes	high (Object-Z)
Harmony	yes	yes	TF	yes	no	low (Z++)

10.3 Main Contributions

The main contributions of our work are the following:

- Pragmatic integration of two notations, one graphical and semi-formal (UML) and the other textual and formal (Z++), in a specification approach that attempts to reap the benefits of both;
- The advanced formalisation of UML constructs in Z++, both in terms of structure and behaviour. It is worth noting that although Lano describes ways of formalising OO models in Z++ [Lano95] this was nevertheless done in the context of the OMT notation

and, while we have not covered all the minute aspects of the formalisation process, our translation from UML to Z++ is performed in a more pragmatic and systematic way, with detailed algorithms being proposed. Also important to note, very few formalisation approaches look at both structure and behaviour, notably [France97] and [Kim00b]), and practically only one within the vicinity of our topic location [RoZeLink99], takes into consideration the reverse propagation, from formal (textual) specifications to semi-formal (diagrammatic) models;

- Rigorous and pragmatic treatment of TCS through the use of a formalism, RTL, whose notation is easy to comprehend and apply. The usability and coverage of our modelling approach stem from its capability of capturing various time-related properties of systems, as discussed in Chapters 5 and 8;
- Lightweight, practical specification process allowing for both reliable specifications and rapid development of software. The main idea of our approach is to provide a rigorous and usable alternative for OO specification of TCS. In order to achieve this, we have focused on the most critical aspects of modelling (in terms of consequences in the life-cycle of the product) and covered the earlier phases of software construction, in particular the OO analysis phase;
- Design of the Harmony ISE, aimed at fully supporting the technique proposed in this thesis. What particularly distinguishes this specification environment is its monolithic construction, support for tight-integration of notations, balanced inclusion of both functions and notational elements (we have attempted to keep things simple, yet still operationally powerful), provisions for easier manipulation of formal symbols, and capacity for extension.

10.4 Other Contributions

There are also a number of aspects of our work that can be listed as secondary contributions. They do not play principal roles in the discourse of this thesis, yet they support it and also represent bits of original work that can be further employed and further investigated:

- Development of a non-trivial example, the ELS. Through this application, which can be added to the rather large collection of elevator case studies recorded in the literature, the most relevant particularities of our approach have been illustrated;
- Classification of integrations of notations. We needed it to provide a basis of comparison with other approaches, but it can be usefully employed or adapted for comparing alternatives of integrating notations in other contexts (e.g., hardware design);
- A zoom-in technique of investigation. The technique has of course been employed in numerous other cases (it is an embodiment of the classical top-down method of investigation), yet there are no reports in literature that present it under the "zoom-in" metaphor;
- Proposal of a class compound construct that encompasses both structure, expressed in the class construct, and behaviour, captured in the state diagram (in addition to the one defined by the operations of the class construct). This pairing of UML constructs (class and state diagram) although quite simple in its idea is nevertheless powerful in that it extends the basic OO concept of encapsulation (data + operations) to a stronger appendage of the type data + operations + allowable sequences of execution;
- Several proposals regarding the terminology: TCS, ISE, tight-integration of notations, monolithic approach, transition signature, transit operation, and the set of terms and abbreviations used to denote the artefacts and steps of our modelling approach;
- A comprehensive review of the research space. Compulsory part of a PhD thesis, of course, but we extended our survey to cover aspects such as UML perspectives and exemplification of UML constructs through the ACTS specification "theme". Both the survey of UML and the modelling of ACTS can evolve in fully-fledged studies on their own rights.

10.5 More On the Limitations of the Proposed Approach

Besides the two main limitations indicated in Section 10.2, namely Z++'s lack of exposure (due primarily to its lack of tools) and Harmony's lack of connection to tools for formal analysis and refinement, there are several other limitations of the proposed approach that

need be addressed in order to enhance the work presented in this dissertation. In particular, the treatment of state diagrams is rather limited, confined to "flat," non-composite structures and to sequential executions, which reduces the applicability of the translation algorithms to modelling TCS (such systems need treatment of concurrency, synchronisation, etc.). Also, the treatment of the timing constraints needs significant improvement, since we have not tackled the automated translation of timing constraints attached to UML structural constructs, and provided only a limited translation of such constraints in the case of state diagrams (the burden of formalising temporal constructs lies too heavily on the human formaliser). In addition, a more concise and precise description of the translation algorithms can be obtained if meta-models for UML and Z++/RTL are used. The deformalisation process also needs improvement; it has been described only by a set of principles and the outline of an algorithm, hence further work on details is needed, as it is needed on dealing with the particular aspects of applying the translations algorithms discussed in Section 6.6.

10.6 A Look Forward

The work presented here is neither complete nor free of errors. We are aware, as indicated in the previous section, of some of its limitations and know that further work is needed in several directions. In particular, our intentions for future work encompass:

- Enhanced automated formalisation and deformalisation. Due to the importance of these processes in producing reliable specifications further studies are necessary, especially regarding the translation of dynamic UML models into precise Z++ specifications;
- Syntax and consistency checking of Z++. We consider the alternative of translating Z++ to Z insufficient, and in order to achieve one of the primary goals of our approach, that of increased intellectual control over the software being developed, automated syntax and consistency checking of formal specifications can play an important role;
- The complete implementation of Harmony. At the time these lines are written, we have completed the design of Harmony. Nevertheless, only by implementing it and exercising

it on various case studies we will be able to both improve its design and gain additional insight about the ways our approach can be efficiently put to work in practice;

- Development of tools for formal analysis and refinement. Although we have not aimed at covering aspects of formal proof and formal refinement, the development of such tools is necessary to support the wider application of our approach;
- More applications. In addition to improving the design of Harmony, the application of the approach on more case studies will be beneficial for fine-tuning the technique of specifying TCS proposed in this thesis.

The above is work that we intend to pursue further in order to develop Harmony into a tool usable on large scale. But, predictably, while working on a given topic ideas for other subjects spring into one's mind, some related and some not so related to the original topic of investigation (and some relatively clear and some decidedly vague). Some of the more related and the slightly more well-formed such ideas that occurred to us are:

- Visualisation of Z constructs in the sense proposed by Kim and Carrington for Z in [Kim99a];
- Usage of the CSP formalism instead of RTL for alternative, enhanced modelling of paralellism;
- Animation of a subset of specifications. A look at solutions such as the Z-based Sum language [Utting95] can provide a starting point;
- Integration of our modelling technique with code generation tools aimed at exploiting the RT capabilities of high-level programming languages;
- Formalisation of modelling patterns and their utilisation in various contexts, for instance for developing Web applications.

At any given time, our Elevator's door may or may not be open depending on a series of factors, as discussed in Chapter 8, but the door of further work and further improvements should always be open.