Chapter 1
Introduction

Prior to postgraduate studies, I worked in a few organisations as a systems developer and a systems administrator[1]. As an on-site systems developer, like many other systems developers, my main duty was to write software for my organisation. But, interestingly, I occasionally found myself involved in other organisational operations which a 'normal' systems developer would not perform (e.g. promoting products in the public space). Equally, the users were sometimes more than just users – they could develop 'simple programs' for themselves. This is particularly true in small and medium sized enterprises (SMEs), where there tends to be a more flexible organisational structure.

Reflection on these personal experiences suggests some interesting questions. On the one hand, it could be argued that fulfilling non-technical roles is part of the systems developer's duty. On the other hand, I may ask "what is the boundary of the systems developer's duties?", "can systems developers in reality only work as 'developers'?" and "can users in reality only work as 'users'?".

Another interesting reflection from my industrial experience is that, despite the fact that there were off-the-shelf (and open source) alternatives, organisations that employed on-site developers tend to build and re-build software of similar kinds over and over again. The interesting question to ask is thus "why would the organisations have to reinvent the wheel over and over again?" One reason is that the organisation process is always situated and evolving at all times. It is unclear how an off-the-shelf component (or system) can be

[1] The difference between a systems developer and a systems administrator is that the former requires knowledge in software construction and the latter requires knowledge in computer hardware. As a systems developer, I mainly wrote software using generic programming languages such as Java and C, and as a systems administrator, I mainly monitored and maintained computer equipment (e.g. servers, routers) using little system-level programming with programming languages such as C and Perl.

customised to suit the organisational needs until the on-site developers and the *real* users get their hands on it. In SMEs, with limited resources to spend, evaluating off-the-shelf components could be a time-consuming activity carrying a degree of risk. This is partially due to the fact that seemingly well-suited off-the-shelf components might prove to be unsuitable after the evaluation, so that the evaluation effort would be wasted, and partially due to the fact that there is no guarantee that off-the-shelf components can be customised to fit new requirements[2]. An economic factor, namely, the cost-to-develop vs. the cost-to-buy-and-customise, could be relevant in this situation. In labour-intensive countries, e.g. China, the labour cost to develop software from scratch is most likely cheaper than the cost of buying an off-the-shelf component and customising it. These considerations all suggest that, to some extent at least, systems development is not merely a technical, but also a socio-technical process.

This thesis studies the development of one particular species of software – groupware – and is motivated by my own experience in groupware development. Groupware development is often a part of a larger systems development project and the developers are working very closely with the other stakeholders (in some cases sharing the same open office space). In contrast to other species of software, groupware is more sensitive to the socio-technical challenge (cf. Grudin, 1994a). Consequently, I argue that groupware development should be seen and treated as an organic process, in which the groupware is grown by the owners with the aid of professional developers rather than constructed by professional developers alone. To realise this perspective, this not only requires the development and proper use of technology, but also harmonious design between the human and the technological. As de Vreede and Guerrero (2006) observed:

> *"… It appears that technology alone seldom is the answer. What is needed is the conscious and harmonious design of collaboration processes and technologies." (ibid, p.571)*

[2] The work practices are more frequently changing in SMEs, especially young SMEs.

1.1 Thesis aim and scope

Empirical Modelling (EM) research was initiated by Meurig Beynon at University of Warwick more than 20 years ago. It has been developed as an approach to computer-based modelling which can be linked to William James's Radical Empiricism – a philosophical foundation that is radically different from that upon which mainstream computer science is based (Beynon, 2005). Over the past two decades, more than 100 peer-reviewed papers and more than a dozen doctoral research theses related to EM have been published. For more details, see chapter 4 and the EM website (http://www.dcs.warwick.ac.uk/modelling).

The aim of this thesis is to show that EM is an alternative and potentially a better conceptual framework for groupware development that accommodates the dynamic nature of group work that the groupware aims to support. More precisely, it tackles three research questions:

i) What are the main challenges in human-centred groupware development?

ii) How can EM be practised in a groupware development context?

iii) Why does EM potentially offer a better conceptual framework?

To answer these questions, I draw on literature on groupware development, human-computer interaction (HCI), computer supported cooperative work (CSCW), participatory design (PD), and Empirical Modelling (EM) in the broader context. In respect of EM, I draw on two strands of previous research:

i) research that concerns EM tool support and the interaction between the modeller and the model, and

ii) research concerned with the potential of EM as a systems development approach (cf. figure 1.1).

Y. W. Yung (1990), Y. P. Yung (1993), P-H. Sun (1999) A. Ward (2004) were the main contributors to the principal tool support (tkeden and dtkeden) for EM. The theses by the brothers Y. W. and Y. P. Yung are both concerned with the low-level differences between EM and other programming paradigms. In particular, Yung (1993) gives a detailed account of the differences in various aspects: *state*, *dependency*, agent, and *definitive notations*.

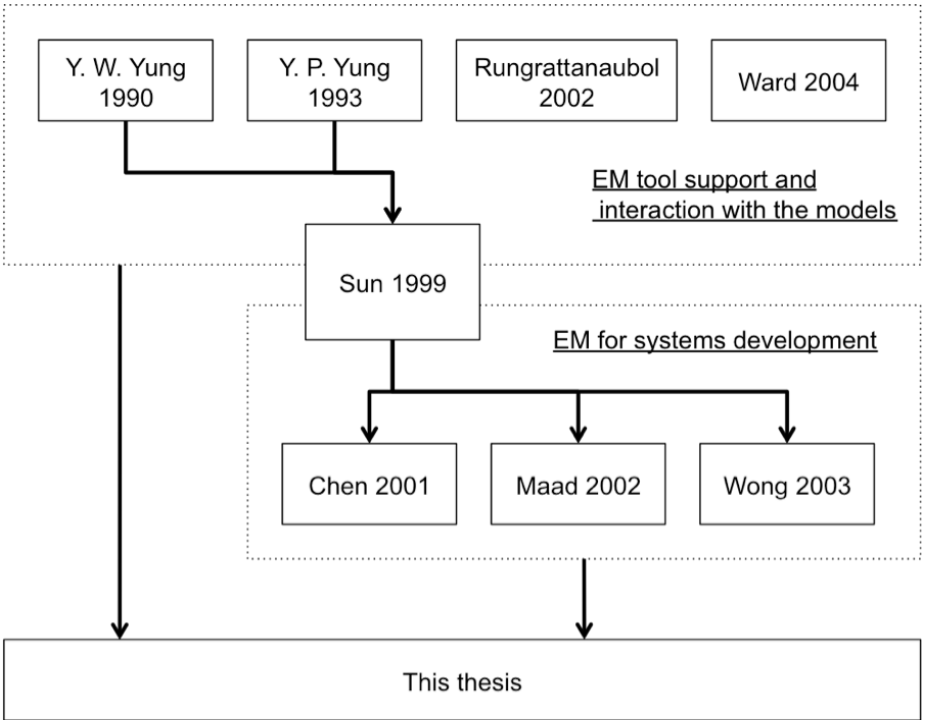

Figure 1.1 – Relationships between this thesis and other doctoral theses on EM

In her doctoral thesis "*A treatise on Modelling with definitive scripts*", Rungrattanaubol (2002) improved our understanding of the core activity of EM: Modelling with Definitive Scripts (MwDS). In particular, she considered in depth how a modeller interacts with states through the (re)definitions of observables, dependencies, agents, and agencies in EM. Ward (2004) explained the difficulty in supporting *dependency* – a key concept in EM (cf. §4.1.1) – on the sequential von Neumann machine hardware.

Sun (1999) introduced a conceptual framework for practising EM in a distributed environment, namely Distributed Empirical Modelling (DEM), in his doctoral thesis: "*Distributed Empirical Modelling and its Applications to Software System Development*". His work improved our understanding the potential of EM in distributed modelling contexts. Since DEM is based on distributed cognition theory (Hutchins, 1995), DEM is evidently more concerned with structured organisations rather than the dynamic structured organisations that might be found in the collaborative modelling context. Sun (1999) also presented a Situated Process of Requirements Engineering (SPORE) to show the potential of the DEM framework. Based on the DEM and the SPORE frameworks, some researchers had shown the potential of practising an EM approach in various domains. For example, Chen (2001)

showed the potential of EM for participative development in business process re-engineering. Maad (2002) showed how the SPORE framework could be used in a financial domain. More recently, Wong (2003) consolidated the EM principles in a Definitive Modelling Framework (DMF). He showed how the developers and the users might interact through the construction of EM models within a SICOD[3] framework in ubiquitous computing.

1.2 Related work

In this section, I briefly discuss a few research works that have a similar outlook with this thesis in the broader context.

The Evolving Artifact Approach

One of the key aspects in groupware development is to co-construct a shared knowledge of the future work practices that will be facilitated by the developing artifact through the development process. Ostwald (1996) argues that the knowledge of future work practices can neither be elicited by observing nor by analysing how the users get their work done within the current work practices. For this reason, Ostwald (1996) proposed the Evolving Artifact Approach (EVA), an evolutionary and participatory approach that "operationalizes the idea of knowledge construction for system development" (ibid). This thesis shares a common interest in the entire lifecycle of systems development (discussed in section 2.5). However, this thesis argues that the users should have equal opportunities to get their hands on co-constructing the artifact within the same modelling space as the developers through EM (discussed in chapter 3, 5, and 7). The philosophical foundation behind EM is rather different from that behind traditional programming. In particular, as will be discussed in chapter 4, EM is based on observation, experimentation, and experience[4].

Meta-design

The idea of viewing systems development as a continual process can also be found in the meta-design conceptual framework (Fischer and Scharff, 2000). Fischer and Scharff (2000)

[3] SICOD stands for "Soft Interfaces for the Control of Devices"

[4] The term 'experience' is used in the sense of William James's (1912) Radical Empiricism.

observe that it is difficult to design a closed system, and assert that, however well such a system is designed, the user will nevertheless discover "mismatches between their problems and the support a system provides". They suggest that the *meta-designer* should *underdesign* (Brand, 1995) the system and let the end-users extend and evolve the system in an open-ended manner, which aligns with the "Seeding, Evolutionary Growth, Reseeding Model" (Fischer et al., 1994). The outlook of the GroupPIE framework (cf. chapter 7) and the efficacious groupware development perspective (cf. chapter 2) are quite similar to the meta-design framework: both evolutionary and participatory. However, "design" in meta-design refers to "[the] activities in which a person wishes to act as an active participant and contributor in personally meaningful activities", which does not necessarily involve the lower level development activities, e.g. system implementation. Therefore, despite the fact that developers and users can be thought as *designers* in the meta-design framework, the imbalance of influence to the artifact construction potentially remain there (cf. chapter 3).

Participatory programming

Research into human-oriented systems development is not new; it was part of the early movement in Participatory Design (PD) (Gasson, 2003; also cf. §2.5.1). While PD, the notion of participation, and the issues surrounding participation will be described in detail in chapter 2 and 3, one particular work in PD is worth mentioning here.

Letondal and Mackay (2004) reported a participatory design project with a group of bioinformatics practitioners who had received professional training in both biology and computer science. The primary aim of their project was to explicitly support end-user programming through PD. The project became interesting as they found two types of bioinformaticians: i) those who maintain a strong interest in biology but are also interested in creating tools for other biologists and themselves to use; ii) those who are "seduced" by the computing tools and are forced to create tools for their problems. This motivated Letondal and Mackay to define the term *participatory programming* as "a logical extension of participation design, in which users participate in the creation of software tools that they can ultimately tailor and program themselves."

Whereas no assumption is made about the technical competence of the participants in EVA

or meta-design, the bioinformaticians in Letondal and Mackay's study were trained to be professional software developers (through their interdisciplinary education). In other words, the bioinformaticians were the developers as well as the users. As the popularity of interdisciplinary programmes, such as the Computer and Business Studies[5] undergraduate course at University of Warwick, increases, we shall see more cases like Letondal and Mackay's. Though there have been some similar findings in other fields, e.g. in free and open source software (FOSS) (Nakakoji et al., 2002) development, only a few researchers (such as (Letondal and Mackay, 2004)) have considered such highly conflated roles in systems development to be possible.

Although in fact EM is not the same as *programming* (cf. chapter 4), *participatory programming* is certainly related to the broader context of this thesis, particularly to the GroupPIE framework that I am going to discuss in chapter 7.

1.3 Research Contribution

The main contribution of this thesis, firstly, is that it identifies a promising configuration for practising an EM approach to groupware development that may lead to *efficacious groupware development*. Secondly, this thesis provides an account of the interplay among the conceptions and issues surrounding participation that leads to a better understanding of the problems in supporting human-centred groupware development. Thirdly, it improves our understanding of the potential of EM in a collaborative modelling context. This is supported with four case studies that cover a range of situations that may frequently occur in a collaborative modelling context. The work provides a promising foundation for further development of EM principles and tool support for collaborative modelling. Lastly, the *efficacious groupware development* perspective is itself a contribution, which provides a unified organic process view of groupware development based on Jullien's interpretation of ancient Chinese philosophy in his book "*The Propensity of Things*".

[5] These students are trained with programming knowledge and business knowledge. They can develop their career in both directions: software development or business management. In the latter case, they may become *hidden professional programmers*.

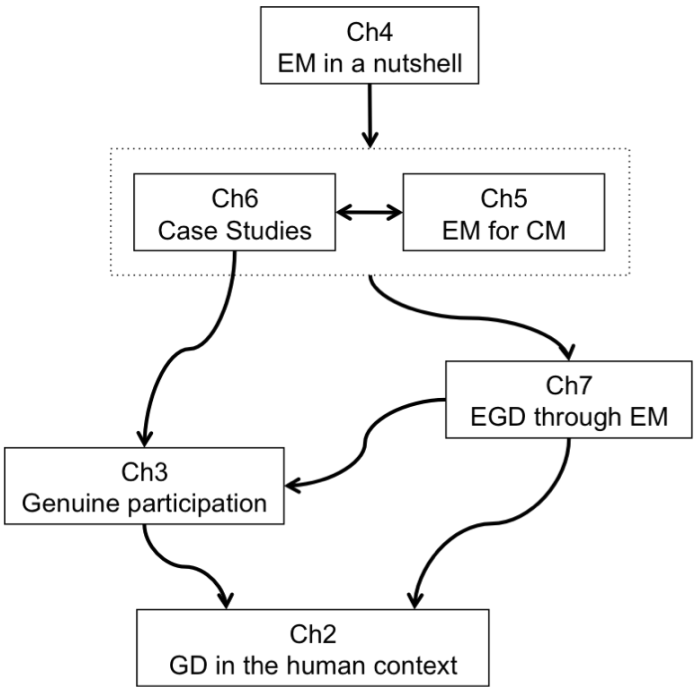

Figure 1.2 – Relationships between chapters in this thesis

1.4 Overview of the thesis

In relation to the aforementioned research questions, this thesis can be divided into three sections (cf. figure 1.2).

The first section (chapter 2 and 3) reviews research into groupware development and argues that several conceptions behind its development are problematic. It answers the question "What are the main challenges in human-centred groupware development?". In chapter 2, I argue that support for co-evolution between the groupware and the group activities is the main issue in integrating groupware into its human context harmoniously. This challenges accepted ideas about the scope of development and the nature of participation. Towards the end of chapter 2, I propose a unified organic process view – the *efficacious groupware development* perspective – that views the development activities as finding an efficacious disposition, which does not distinguish contexts and roles of participants throughout the development and evolution of groupware. In chapter 3, I argue that the contexts of development and use, the roles of participants, and the nature of participation are interdependent concepts. Instead of re-conceptualising them, I argue for the support of *genuine participation* which conflates contexts and roles, and see participants

as human-beings. This creates a conceptual orientation for the rest of the thesis.

The second section (chapter 4, 5, and 6) discusses EM principles and tools, and explores EM potential in collaborative modelling. It answers the questions “What is EM?” and “How can EM be practised in a collaborative modelling context?”. EM is a modelling approach which is based on experience, interaction, observation, and experimentation. In chapter 4, I explain the principles, the philosophical outlook, and the practical EM tool support. Drawing on various previous research into EM (cf. figure 1.1), I argue that the EM process is open and flexible, and that EM artifacts (or models) are provisional, situated, tacit, interactive, and ostensive (in the sense of Gooding’s construal). In chapter 5, I define what I mean by *collaborative modelling*. I argue that collaborative modelling involves diverse configurations that can neither be prescribed nor framed easily. A better approach to support collaborative modelling is to take this heterogeneous nature into account and allow flexible reconfiguration. I also examine the potential of EM in a collaborative modelling context, and explain why DEM is not good enough for collaborative modelling. In chapter 6, I demonstrate the potential for practising an EM approach in the context of collaborative modelling with four case studies. These case studies cover situations that may occur frequently in the context of collaborative modelling.

The last section (chapter 7) considers EM in the context of groupware development. It answers the questions “Why does EM potentially offer a better conceptual framework?” and “How can EM be practised in a groupware development context?. Based on the principles of EM and the case studies, I argue that EM is a human-centred development approach when it is practised in a collaborative modelling context. Therefore, I argue that the potential of EM in collaborative modelling can be exploited in groupware development. Such an orientation potentially yields a better conceptual framework that gives a holistic account of groupware development from its first conception to its eventual disposal. This is a step towards realising efficacious groupware development.