On Utility-based Selection of Architecture-Modelling Concepts

H.A. Proper[1], A.A. Verrijn-Stuart[2] and S.J.B.A. Hoppenbrouwers[1]

[1]Radboud University Nijmegen, Sub-faculty of Informatics, IRIS Group
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU.
e.proper@acm.org, stijnh@cs.kun.nl
[2]Emeritus of the University of Leiden, Faculty of Mathematics and Science, LIACS
Scheltemakade 15 (home), 2012 TD Haarlem, The Netherlands, EU.

On the 26th of October, Xander Verrijn-Stuart passed away unexpectedly.
We dedicate this paper to Xander, and hope his memories may continue to inspire us in the years to come.

PUBLISHED AS:

H.A. Proper, A.A. Verrijn-Stuart, and S.J.B.A. Hoppenbrouwers. Towards utility-based selection of architecture-modelling concepts. Technical Report NIII-R0417, Nijmegen Institute for Information and Computing Sciences, University of Nijmegen, Nijmegen, The Netherlands, EU, 2004.

Abstract

In this position paper we are concerned with the principles underlying the utility of modelling concepts, in particular in the context of architecture modelling. First, some basic concepts are discussed, in particular the relation between information, language, and modelling. Our primary area of application is the modelling of enterprise architectures and information system architectures, where the selection of the concepts used to model different aspects very much depends on the specific concerns that need to be addressed. We present an approach to utility-based concept selection. Our approach is illustrated by a brief review of the relevant aspects of two existing frameworks for modelling of (software intensive) information systems and their architectures. We finish this paper by a discussion of our future research directions in this field.

Keywords: MODELLING CONCEPTS, MODELLING LANGUAGES, ARCHITECTURE MODELLING

1 Introduction

The importance of information systems to modern day society needs no arguing. Information systems may range from small-scale systems geared towards a few users, via systems supporting the tasks of a business unit, to enterprise-wide systems and even value-chain wide systems. The ubiquity of information systems, combined with their far-reaching integration with our daily lives (both at work and at home) puts high demands on the development processes that bring forth these systems.

Information systems, as their name suggests, primarily handle 'information'. Based on the definition provided in [Falkenberg et al., 1998], we define *information* as "the *knowledge* increment brought about when an *actor* receives a message". As a direct consequence, we regard messages that result from an information system as representations of knowledge. In line with [Falkenberg et al., 1998, Bernus et al., 1998] we consider an *information system* to be a system for collecting, processing, storing, retrieving and distributing information within an organisation and between the organisation and its environment. As such, an information system can be regarded as a subsystem of the organisation (focussing on the informational aspects of an organisation), and may consist of both *human* and *computerised* actors.

In the last decennium, several approaches to the development of larger information systems (anything beyond small scale systems geared towards a few specific users) have emerged that strongly depend on the use of so-called '*architectures*' [Zachman, 1987, Boar, 1999, Bernus et al., 1998]. Some of these approaches use the term 'information architecture', or 'information systems architecture', while yet others refer to the same concept as 'enterprise (IT) architecture'.

In [IEEE, 2000], the concept of architecture is defined as: "*The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.*". Architectures are usually expressed in terms of architectural descriptions, essentially design descriptions pertaining to the architecture of a system. In general, the rationale behind the use of architecture in the context of information systems is that it provides a number of important benefits [Bass et al., 1998, IEEE, 2000], such as:

- It is a vehicle for communication and negotiation among stakeholders. A software architecture, often depicted graphically, can be communicated with different stakeholders involved in the development, production, fielding, operation, and maintenance of a system.
- It captures essential design decisions, both functional aspects as well as quality aspects. In an architecture, the global structure of the system has been decided upon, while responsibilities (such as functionality) have been assigned to the (overall) components of the system.

Part of this work was conducted within the ArchiMate project (http://archimate.telin.nl), a research initiative that aims to provide concepts and techniques to support enterprise architects in the visualisation, communication and analysis of integrated architectures. The ArchiMate consortium consists of ABN AMRO, Stichting Pensioenfonds ABP, the Dutch Tax and Customs Administration, Ordina, Telematics Institute, CWI (Centre for Mathematics and Informatics), Radboud University Nijmegen, and the Leiden Institute of Advanced Computer Science.

In the conceptual framework for architecture as defined in [IEEE, 2000], an architectural description can be organised into one or more constituents called architectural views. Each view addresses one or more of the *concerns* (interests) of the stakeholders of a system. The term 'view' is used to refer to the expression of a system's architecture with respect to a particular viewpoint. A viewpoint establishes the conventions by which a view is created, depicted and analyzed. In other words, a viewpoint determines the languages to be used to describe the view, and any associated modelling methods or analysis techniques to be applied to these representations of the view. These languages and techniques are used to yield results relevant to the concerns addressed by the viewpoint.

The concept of viewpoint is not new. For example, Multiview [Wood-Harper et al., 1985] already introduced the notion of views. Multiview identifies five viewpoints: Human Activity System, Information Modelling, Socio-Technical System, Human-Computer Interface and the Technical System. During the same period when Multiview was developed, the so-called CRIS Task Group of the IFIP working group 8.1 developed similar notions, where stakeholder *views* were reconciled via appropriate "representations" and special attention was paid to the then common disagreement about which aspect (or *perspective*) was to dominate system design (viz. "process", "data" or "behaviour"). As a precursor to the notion of *concern*, the CRIS Task Group identified several *human roles* involved in information system development, such as *executive responsible*, *development coordinator*, *buiness analyst*, *business designer*, etc. The results of that work can be found in a book entitled "*Information System Methodologies: A framework for understanding*" [Olle et al., 1988a] and in the proceedings of the CRIS conferences from 1982–1991 [Olle et al., 1982, Olle et al., 1983, Olle et al., 1986, Olle et al., 1988b, Verrijn-Stuart and Olle, 1991].

The use of viewpoints is not limited to the information systems community, it was also introduced by the software engineering community. In the 1990's, a substantial number of software engineering researchers worked on what was phrased as "the multiple perspectives problem" [Finkelstein et al., 1992, Kotonya and Sommerville, 1992,
Reeves et al., 1995]. By this term, the authors referred to the problem of how to organise and guide (software) development in a setting with many actors, using diverse representation schemes, having diverse domain knowledge, and using different development strategies. A general framework has been developed in order to address the diverse issues related to this problem [Finkelstein et al., 1992, Kotonya and Sommerville, 1992]. In this framework, a viewpoint combines the notion of *actor*, *role*, or *agent* in the development process with the idea of a *perspective* or *view* which an actor maintains. A viewpoint is more than a *partial specification*; in addition, it contains partial knowledge of how to further *develop* that partial specification. These early ideas on viewpoint-oriented software engineering have found their way into the IEEE-1471 standard for architectural description [IEEE, 2000] on which we have based our definitions below.

In the context of architectural descriptions, a plethora of frameworks of viewpoints exists. Some of these frameworks of viewpoints are: The Zachman framework [Zachman, 1987], Kruchten's 4+1 framework [Kruchten, 1995], RM-ODP [ISO, 1998], ArchiMate [Jonkers et al., 2003] and TOGAF [TOGAF, 2004]. The aim of this position paper is not to provide 'yet another framework of viewpoints', but rather to argue the need for a fundamental approach to the selection of viewpoints (and modelling techniques in general), as well as to lay a foundation that enables us to *reason about such frameworks at a meta-level*. In other words, we seek a fundamental approach allowing designers and architects to consider the relevance of specific viewpoints regarding their practical design/development tasks. Instead of making superficial comparisons between the specific abilities of various techniques, we aim at finding deeper motivations for the differences between them.

This paper is a product of ongoing research which aims to gain a more fundamental understanding of the act of modelling in the context of system development, and the languages that are used in the process. The view presented is developed via three complementary angles:

Modelling: This angle aims to provide a fundamental grounding of (architectural) modelling and representation. We will focus primarily on the foundations of modelling, representation of models and the role of languages.

Utility: The potential utility that specific modelling concepts may have when used to express architectural descriptions, from the perspective of a given design/development task.

This angle, which builds on the previous one, aims to provide designers and architects with the insights to reason about the relevance of modelling concepts to a specific task in the design/development process.

Communication: The (interpersonal) communication about architectural descriptions as it occurs during modelling and design.

This angle, further adding to the insights of the previous two, focuses on the role of architectural descriptions as a means of communication between a system's stakeholders, i.e. language in action.

Thus far, our research in this area was directed primarily at the modelling of architectures and the selection of suitable viewpoints, in the context of the ArchiMate project [Jonkers et al., 2003, Jonkers et al., 2004]. The results, however, can equally well be applied to the selection of modelling techniques in general. In both the past and present, a number of information systems modelling techniques have been, and are being, developed [Bubenko, 1986, Avison and Wood-Harper, 1991, Avison, 1995, Bernus et al., 1998]. The authors of this paper have themselves contributed their fair share of modelling techniques [Bommel et al., 1991, Hofstede and Weide, 1993, Bronts et al., 1995, Proper and Weide, 1994, Creasy and Proper, 1996, Campbell et al., 1996, Hoppenbrouwers et al., 1997]. The myriad of modelling techniques that is available to developers of information system has, in the past, already been referred to as "a jungle" [Avison, 1995]. The viewpoints of the different architecture framework contribute even more techniques to this jungle, burdening architects/developers with the selection of appropriate modelling techniques for the modelling tasks at hand.

We have structured the remainder of this paper as a discussion of the three angles mentioned above, with a section each (sections 2 to 4). To make our results more concrete, in section 5 we discuss two example frameworks of viewpoints (Kruchten's 4+1 framework [Kruchten, 1995] and RM-ODP [ISO, 1998]), from the perspective of our meta-framework. Before

concluding, section 6 provides a brief discussion on the perspective on information system development we will take in further development of our theory.

2 Modelling

The aim of this section is to closely investigate the process of modelling as it occurs in, for example, architecture-modelling. In defining more precisely what we mean by modelling a domain, we first need to introduce a framework describing the essential processes that take place when someone (a stakeholder for example) observes a domain (such as a system being developed).

Let us first consider what happens if some *viewer* observes 'the universe'. Following C.S. Peirce [Peirce, 1969a, Peirce, 1969b], we assume that viewers perceive the universe and then produce a *conception* of that part of it they deem relevant. The conceptions harboured by a viewer cannot be communicated and discussed with other viewers unless they are articulated somehow (the need for this ability in the context of system development is evident). In other words, a *conception* needs to be *represented*. Peirce argues that both the perception and conception of a viewer are strongly influenced by their interest in the observed universe. This leads to the following (necessarily cyclic, yet irreflexive) set of definitions:

Universe – the 'world' around the viewer.

Viewer – an actor perceiving and conceiving the universe, using their senses.

Conception – that which results, in the mind of a viewer, when they observe the universe, using their senses, and interpret what they perceive.

Representation – the result of a viewer denoting a conception, using some language and medium to express themselves.

The underlying relationships between viewers, universe, conceptions and representations can be expressed in terms of the so-called FRISCO tetrahedron [Falkenberg et al., 1998], as depicted in figure 1.

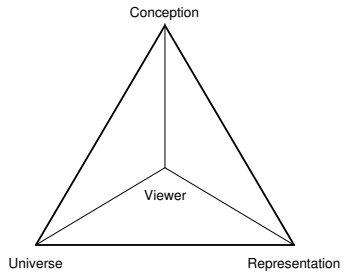

Figure 1: The FRISCO tetahedron.

As mentioned above, in conceiving a part of the universe, viewers will be influenced by their particular interest in the observed universe. In the context of system development, this corresponds to what tends to be referred to as a *concern*. For example:

- The current situation with regard to the computerised support of a business process.
- The requirements of a specific stakeholder with regard to the desired situation.
- The potential impact of a new system on the work of the system administrators that are to maintain the new system.

Concerns are not the only factor that influences a viewer's conception of a domain. Another important factor is the pre-conceptions viewers may harbour, brought forward by their social, cultural, educational and professional background. More specifically, in the context of architecture modelling, viewers will approach a domain with the aim of expressing it in terms of some set of meta-concepts, such as classes, activities, constraints, etc. The set of meta-concepts a viewer is used to using (or trained to use) when modelling a domain will strongly influence the conception of the viewer. This can be likened to the typical situation of having a 'hammer' and considering all pointy objects to be 'nails'. We therefore presume that when viewers model a domain, they do so from a certain perspective; their *weltanschauung* (German for "view of the world") [Wood-Harper et al., 1985]. The Weltanschauung can essentially be equated to the notion of a *viewpoint* as discussed in section 1. This perspective on the notion of viewpoints is compatible to the approach taken in the Reference Model of Open Distributed Processing [ISO, 1998]:

> *"In order to represent an ODP system from a particular viewpoint it is necessary to define a structured set of concepts [the meta-concepts] in terms of which that representation (or specification) can be expressed. This set of concepts provides a language for writing specifications of systems from that viewpoint, and such a specification constitutes a model of a system in terms of the concepts."*

In general, people tend to think of the universe (the 'world around us') as consisting of related *elements*. In our view, however, presuming that the universe consists of a set of elements already constitutes a subjective choice, which essentially depends on the viewer observing the universe. The choice being made is that 'elements' (or 'things') and 'relations' are the most basic concept for modelling the universe; the most basic Weltanschauung. In the remainder of this paper, we will indeed make this assumption, and presume that a viewer's *conception* of the universe consists of elements. The identification of elements in the universe remains relative to viewers and their *own* conception.

Viewers may decide to zoom in on a particular part of the universe they observe, or to state it more precisely, they may zoom in on a particular part of *their* conception of the universe. This allows us to define the notion of a domain as:

Domain – any subset of a conception (being a set of elements) of the universe, that is conceived of as being some 'part' or 'aspect' of the universe.

In the context of (information) system development, we have a particular interest in unambiguous abstractions from domains. This is what we refer to as a *model*:

Model – a purposely abstracted and unambiguous conception of a domain.

Note that both the domain and its model are *conceptions* harboured by the same viewer. We are now in a position to define more precisely what we mean by modelling:

Modelling – The act of purposely abstracting a model from (what is conceived to be) a part of the universe.

For practical reasons, we will understand the act of *modelling* to also include the activities involved in the *representation* of the model by means of some language and medium.

We presume viewers not only to be able to represent (parts of) their conceptions of the universe, but also to be able to represent (parts of) the viewpoints they use in producing their conception of the universe. This does require viewers to be able to perform some kind of self-reflection. When modelling some domain in terms of, say, UML class diagrams [Booch et al., 1999], viewers/modellers are presumed to be able to express the fact that they are using classes, aggregations, associations, etc, to view the domain being modelled. In doing so, viewers essentially need to construct a conception of their viewpoint on the world; i.e. a meta-model. This meta-model comprises the meta-concepts and modelling approach used by the viewer when modelling a domain; it is a model of the viewers viewpoint. Such a meta-model can in essence be regarded as a 'high level ontology' [Kishore et al., 2004].

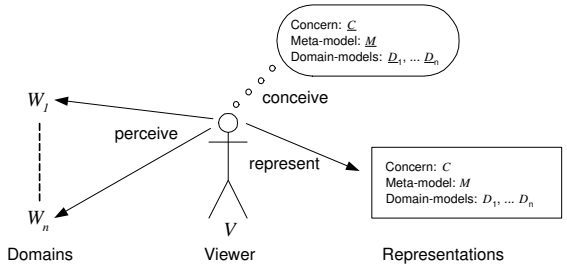

Figure 2: A viewer viewing domains from a particular concern and meta-model.

In figure 2 we have depicted a situation where a viewer is confronted with a number of domains $(W_1, \ldots, W_n)$. Each of these domains may be modelled from the perspective of the viewer's concern $\underline{C}$ and meta-model $\underline{M}$, leading to as many domain-models $(\underline{D}_1, \ldots, \underline{D}_n)$. The concern, the meta-model, and the domain models can be represented using some language and medium, leading to representations $C, M, D_1, \ldots, D_n$.

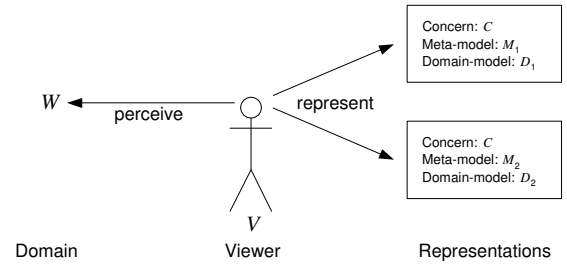

Figure 3: A viewer viewing a domain from the perspective of two different meta-models.

A viewer may also consider a specific domain W from the perspective of some concern C, using two different meta-models M_1 and M_2. This situation is illustrated in figure 3, where a viewer models a domain D from the perspective of meta-models M_1 and M_2, leading to domain-models D_1 and D_2 respectively. For example, when viewing a domain from the perspective of a UML class diagram, this is bound to lead to a different domain-model than the same domain being viewed from the perspective of a UML sequence diagram.

If a viewer observes a domain D using the same meta-model M, but from the perspective of different concerns C_1 and C_2, it is also quite likely that the viewer will produce different domain-models, each catering to the specific needs of the respective concerns. Consider, for example, a concern focussing on the functionality offered by a system to its users, versus a concern focussing on the impact of the system on the efficiency of business processes.

Given two different concerns, it is likely that questions underlying them cannot be met by using a one-size-fits-all meta-model. For example, the operators who will be required to maintain a planned information system will regard this system in terms of costs of keeping the system up and running, costs and efforts involved in implementing the system, etc. Future users of the same planned system, however, will be more interested in the impact/support the system is likely to have on their work related tasks. This implies that when modelling a system (being designed/developed), different meta-models need to be used to address different concerns.

The aim of the next section is to gain more insight into the potential utility of a meta-concepts to a given modelling goal and viewer's concern.

3 Utility of modelling concepts

A fundamental problem to be addressed is the 'utility' of modelling concepts relative to some concern and modelling goal. Utility must be understood in the sense of its classical economic context, such as "what benefit do I derive from using it?" or "of what use is it to me?". Our primary concern is with the area of information systems, of which both the modelling and model usage aspects must be considered. The former pertain to the expressiveness of modelling languages, the latter to the effectiveness of the ultimate system.

3.1 Representational economics

Intuitively, information is linked to knowledge (knowing how to do something, how to do it better, how to do it more timely, or how to avoid something). In other words, information is an important ingredient of decision making. As mentioned above, a sensible definition is to equate 'information' to an increment in knowledge [Falkenberg et al., 1998]. Value of information (or its 'utility') should then be associated with the advantage of better decision making and more effective 'goal-pursuit' [Falkenberg et al., 1998, Section 2.5 and Chapter 3].

It is evident that the 'economics of information' are complex and cannot be derived from – if at all associated with – some identifiable and coherent market. While everyone will agree that something referred to as 'information' must have value, the contexts will vary so much that any pretense to a straightforward theory should be rejected. One of the reasons is that what we actually deal with are *representations* of domains in the real or imaginary world (i.e. 'models'), and representations of things, relationships and actions pertaining to those domains (i.e. 'data'). The theme of this paper is the utility of *how* and *what* one models. Regarding the link with the economics of information we will merely assert three things:

- a computerised system capable of providing *information* is of *value* to its owner/user;
- the *cost* of building and maintaining such a system may exceed its *value addition*;
- any means of *reducing cost* and/or *increasing value* in this context are desirable.

Since the systems we wish to construct are dynamic representations of a domain of interest, a key question is how best to describe such domains. What we

look for is a meta-model which is simultaneously simple and rich. For instance, it must be capable of describing anything requiring modelling in our domain. On the other hand, it must be so restricted that the full range of concepts may be grasped by any individual modeller while being simultaneously usable in exchanges between such a person and the many other interested parties concerned.

The solution to this problem lies in a judicious selection of 'concepts'. These should be both operationally effective and domain-encompassing. Effectiveness means having a high 'utility', that is to say, they must be generally accepted and permit 'economical' application. Encompassing one particular type of domain, but not necessarily all conceivable ones, means that a 'goal-bounded' approach should be adopted whereby the modelling needs are restricted according to the modelling goal at hand. The modelling goal may differ from situation to situation. The problem, therefore, evolves to that of agreeing on and maintaining a large collection of concepts suitable to cover a range of domains and a systematic means of bounding it to fit any selected occasion.

The best way of achieving this is to resort to a set of high-level 'meta-concepts', which can be specialised to fit contingencies. How general and large should that set be, what should be in it and, most importantly, how do we effectively and economically restrict it to cope with specific domains?

3.2 Meta-concepts and concept restrictions

One way of dealing with complexity of a given situation is to abstract its features and describe it in general terms. For instance, if an extended set of concepts is required, then grouping them in broad categories aids better understanding. Similarly, when in spite of all simplification efforts a complex language remains necessary, discussing and describing it in a 'meta-language' is often fruitful.

Thus, the problem of arriving at a 'goal-boundable' approach to modelling is to conceive an extremely simple *set of meta-concepts*, each of which is capable of being specialised so as to be used in a particular case. Information systems – which serve organizations – have two general aspects, both of which need to be captured in any model:

- Informational aspect – i.e. 'what to describe', but also, 'what to leave out'.
- System aspect – i.e. a cohesion-oriented 'description format'.

While the first gives rise to the *elements* of the ultimate system, the latter provides the *formal basis* for putting them together. An 'information system' is a special kind of 'system' and a 'system', in turn, is a specialisation of a 'model' [Falkenberg et al., 1998]. Therefore, the concepts appropriate for underpinning an information system description must, after all, derive from the most general concepts for modelling. These also constitute our desired set of 'meta-concepts'. The information and system aspects characterise the special domain 'information system' and, hence, may be covered under the umbrella 'domain-concepts' (covering all elementary and structural features of information systems, as such).

To summarize, our aim must be to devise a complete – and ideally minimal – set of *meta concepts* that are specific to the field of *information systems* and to develop a well founded *procedure for specialising* this set, in a utility-driven and goal-bounded way, according to the requirements of specific situations pertaining to the development and evolution of information systems.

4 Utility of Concepts in Communication and Computation

Let us now consider a functional or *utilitarian* view on concepts as they are integrated in some *language* that is being used to communicate. This boils down to the question: 'what is it that concepts are *for*'?

4.1 A utilitarian view on concepts and meta-concepts

Roughly speaking, there are two main areas of use for concepts: *communication* and *computation*. Though these uses are often heavily entwined – to the point where they can hardly be distinguished – at a more fundamental level of analysis they are completely different [Hoppenbrouwers, 2003]. The distinction between communication- and computation-oriented concept use is related to the two general aspects discussed in section 3: the 'informational aspect' and the 'system aspect'.

Concepts for communication are bound up with languages in order to communicate: exchange information, and thereby ultimately change the knowledge of some individual in line with some intention of another individual to do so. Such concept use is very strongly tied up with communication between humans as studied in linguistics and communication theory. The utility of concepts for communicative use is therefore related to *principles of effective communication*.

Concepts for computation form *symbolic structures* that are usually intended as part of an engineered artefact. Even though their status is not primarily 'physical', there is essentially a clear and unambiguous link between the symbols in the structure (for example, programming code) and an underlying piece of hardware. This comes clearly to the fore in the case of assembly code, microcode and hardware, in particular in the trade-off between the realisation of computational functionality in hardware or microcode. The computational use of concepts is mostly tied up with fields like electronic engineering, computer engineering and software engineering. The utility of concepts for computational use is therefore related to *principles of good engineering*.

Ideally, a balance would need to be struck in each situation between the sorts of utility involved. Architects are perhaps the most important group of people that carry the burden of bridging the gaps between levels, layers, groups, and activities involved in information system development.

In system development, both concepts for communication and concepts for computation are subject of continuous discussion. Such 'meta-conversations' [Hoppenbrouwers, 2003] may concern the labelling of a concept (typically, the word form associated with it in some language), or the meaning of a concept. However, the very idea of (how to discuss and represent) 'meaning' is usually quite different in the two areas of use distinguished.

In order to avoid the discussion concerning "the meaning of meaning" [Putnam, 1975], it is possible to take a strictly functional (i.e. *utilitarian*) approach to conversation about concepts. In that case, we look upon communication about concepts as striving for sharing knowledge about the meaning of a concept (according to whatever view on 'meaning' is deemed relevant) so that parties involved can, in their own opinion, effectively communicate.

The contexts for meta-conversation may differ quite substantially. In particular, the difference between meta-conversation in context of 'concept use for communication' will often be radically different from that in context of 'concept use for computation'.

4.2 Conversations for conceptual mediation

There is a third type of utility for concepts. In the various disciplines and activities involved in large-scale information system development, a multitude of terms, concepts, and languages is used, combining the two main utilities in many different ways. Straightforward strategies to deal with this (assuming involvement of only two parties for sake of the argument) are the following:

- One party acquires/uses the vocabulary of the other.
- Both parties acquire/use each other's vocabulary.
- Both parties acquire/use a third vocabulary, a *lingua franca*.
- A third party steps in as a *translator*.

Note that these strategies apply to both types of utility in concept use, and also *between them*. However, the 'conceptual mediation utility of concepts' is especially closely related to meta-conversation. Strictly speaking, it may not only involve an active combination of *various* meta-languages and meta-conversation strategies, but even a class of meta-language/strategies of its own: meta-language specific to the task of bridging gaps between languages or (types of) meaning description. (For example, consider conversations between *translators*).

Architects, more than any other professional group, should be able to be active conceptual mediators. Modelling concepts, and the various meta-languages and conversation strategies that may be used to discuss them and align their meaning between parties, are in the center of the utilitarian (meta-)discussion because they are particularly vulnerable to confusion and incompatibility resulting from the communication-computation opposition, and related mix-ups.

5 Case Studies

This section aims to illustrate the above discussed principles by briefly positioning two existing frameworks of viewpoints. In the discussions above, we have argued how the combination of a goal and a stakeholder concern should ideally dominate the choice of a specific meta-model when modelling a domain. The motivation for specific choices of concepts in the meta-model should be formulated in terms of the utility these concepts may bring towards the modelling goal and stakeholder concern. This utility may pertain to the potential communicative, computational as well as mediative use of the concepts.

In the case studies we will primarily focus on the viewpoints identified, the goals and concerns they aim to serve, the concepts in the associated meta-model, and any motivations for the specific concepts in the meta-model. The two case studies are not intended as criticism on the original frameworks. They may, however, indicate that some important considerations (such as utility based motivations) have been left out of the (original) publication of the frameworks. This does not imply that the framework as such is at fault, but rather that interesting and important motivations underlying the specific frameworks have been left implicit. Furthermore, the case studies are part of an ongoing research effort. More detailed studies of the cases will be presented in later work.

5.1 The '4+1' view model

In [Kruchten, 1995], Kruchten introduces a framework of viewpoints (a view model) comprising five viewpoints. The use of multiple viewpoints is motivated by the observation that it "*allows to address separately the concerns of the various stakeholders of the architecture: end-user, developers, systems engineers, project managers, etc., and to handle separately the functional and non-functional requirements*". Kruchten does not explicitly document the motivation for these specific five viewpoints. This also applies to the version of the framework as it appears in [Kruchten, 2000, Booch et al., 1999].

The goals, stakeholders, concerns, and meta-model of the 4+1 framework can be presented, in brief, as shown in table 1. Note that in [Kruchten, 2000, Booch et al., 1999], the viewpoints have been renamed to better match the terminology of the UML: physical viewpoint → deployment viewpoint, development viewpoint → implementation viewpoint and scenario viewpoint → use-case viewpoint.

The framework proposes modelling concepts (the meta-model) for each of the specific viewpoints. It does so, however, without explicitly discussing how these modelling concepts indeed contribute towards the goals of the specific viewpoints. Are, for example, object-classes, associations, etc, the right concepts to communicate with end-users about the services they require from the system? The 4+1 framework is based on experiences in practical settings by its author. This should make it even more interesting to make explicit the motivations, in terms of utility, for selecting the different modelling concepts. Unforunately, in [Kruchten, 2000, Booch et al., 1999] this is also not documented. The viewpoints are presented 'as is'.

5.2 RM-ODP

The Reference Model of Open Distributed Processing (RM-ODP) [ISO, 1998] was produced in a joint effort by the international standard bodies ISO and ITU-T in order to develop a coordinating framework for the standardisation of open distributed processing. The resulting framework defines five viewpoints: *enterprise*, *information*, *computation*, *engineering* and *technology*. The modelling concepts used in each of these views are based on the object-oriented paradigm. The goals, concerns, and associated meta-models of the viewpoints identified by the RM-ODP can be presented, in brief, as in table 2.

The RM-ODP provides a modelling language for each of the viewpoints identified. Furthermore it states:

> *"Each language [for creating views/models conform a viewpoint] has sufficient expressive power to specify an ODP function, application or policy from the corresponding viewpoint."*

The authors fail to provide a more detailed discussion regarding the utility of the concepts underlying each of these languages, from the perspective of the goals/concerns that are addressed by each of the viewpoints. Also, the RM-ODP does not explicitly associate viewpoints to a specific class of stakeholders. This can only be derived from the link with the concerns which the viewpoints address.

In particular in the case of an international standard, it would have been interesting to see explicit motivations, in terms of utility to the different goals, for the modelling concepts selected in each of the views.

5.3 Summary

Neither of the viewpoint frameworks discussed fail to provide an explicit motivation for their choices re-

Viewpoint:	Logical	Process	Development	Physical	Scenarios
Goal:	Capture the services which the system should provide	Capture concurrency and sychronisation aspects of the design	Describe static organisation of the software and its development	Describe mapping of software onto hardware, and its distribution	Provide a driver to discover key elements in design Validation and illustration
Stakeholders:	Architect End-users	Architect System designer Integrator	Architect Developer Manager	Architect System designer	Architect End-users Developer
Concerns:	Functionality	Performance Availability Fault tolerance ...	Organisation Re-use Portability ...	Scalability Performance Availability ...	Understandability
Meta-model:	Object-classes Associations Inheritance ...	Event Message Broadcast ...	Module Subsystem Layer ...	Processor Device Bandwidth ...	Objects-classes Events Steps ...

Table 1: Classification of the 4+1 viewpoint framework.

Viewpoint:	Enterprise	Information	Computational	Engineering	Technology
Goal:	Capture purpose, scope and policies of the system	Capture semantics of information and processing performed by the system	Express distribution of the system into interacting objects	Describe design of distribution oriented aspects of the system	Describe choice of technology used in the system
Concerns:	Organisational requirements and structure	Information and processing required	Distribution of system Functional decomposition	Distribution of the system, and mechanisms and functions needed	Hardware and software choices Compliancy to other views
Meta-model:	Objects Communities Permissions Obligations Contract ...	Object classes Associations Process ...	Objects Interfaces Interaction Activities ...	Objects Channels Node Capsule Cluster ...	Not stated explicitly

Table 2: Classification of the RM-ODP viewpoint framework.

garding the modelling concepts used in specific viewpoints. When using one of the frameworks, architects will not find it difficult to select a viewpoint for a given modelling task at hand. However, this 'ease of choice' is more a result of the limitation of options available (one is limited to the number of viewpoints provided by the framework) than the result of a well motivated choice about the viewpoint's utility towards the tasks at hand. Even more, making a choice for one of the two frameworks (or rather, one of the many, many, other frameworks) will be hard on rational grounds, without such utility based motivations.

6 Outlook: Understanding Requirements on Modelling Techniques

This position paper states the need for a utilitarian view on modelling techniques and modelling concepts. Thus far, we have done so without providing a framework reference in terms of which this utility should be determined. Relative to what *requirements* should the utility be judged? This section discusses our fundamental way of thinking with regards to system development, and as such provides a frame of thought within which one can evaluate the potential utility of modelling concepts/techniques. It is this frame of thought which we shall apply in further development of our theory.

6.1 Communication-driven

Key to our view on the utility of modelling techniques is their role as a means of communication in system development. In the past we have already taken a communication-driven perspective on modelling activities in information system development [Derksen et al., 1996, Hoppenbrouwers et al., 1997, Frederiks and Weide, 2004, Frederiks and Weide, 2004, Bleeker et al., 2004, Proper and Hoppenbrouwers, 2004], as well as the act of system development itself [Veldhuijzen van Zanten et al., 2004]. We are certainly not not alone in doing so [Nijssen and Halpin, 1989, Embley et al., 1992, Halpin, 2001].

To better understand the role of modelling techniques in system development, we have extended our communicative perspective to cater for the fact that the communication taking place during during system development leads to the creation and dissemination of *knowledge*. In essence, we will regard system development as a communication-driven *knowledge transformation* process whereby conversations are used to share and create knowledge pertaining to both the system being developed, as well as to the development process itself. The notion of conversation should be interpreted here in the broadest sense, ranging from a single person producing a model (description), via a one-on-one design/elicitation session, to a workshop with several stakeholders and even the widespread dissemination of definitive system designs.

We also do *not* claim that viewing information system development as a knowledge transformation process is new [Mylopoulos, 1998]. Our aim is to use this perspective on system development to better understand and articulate the requirements that underly modelling techniques. From this perspective, modelling techniques should be regarded as a means (a language) to an end (system development), not unlike a functional (*What is it to be used for?*) perspective on language [Cruse, 2000].

6.2 Development community

Given a focus on communication, it is important to identify the actors and objects that could play a role

in the communication that takes place during the system development process. These actors are likely to have some stake with regards to the system being developed. Examples of such actors are: problem owners, prospective actors in the future system (such as the future 'users' of the system), domain experts, sponsors, architects, engineers, business analysts, etc.

These actors, however, are not the only 'objects' playing an important role in system development. Another important class of objects are the many different documents, models, forms, etc., that *represent* bits and pieces of knowledge pertaining to the system that is being developed. This entire group of objects, and the different roles they can play, is what we shall refer to as a *system development community*:

> *a group of objects, such as actors, models and representations, which are involved in the development of a system.*

The *actors* in a system development community will (typically as a consequence of their personal *goals* and *stakes*) have some specific interests with regards to the system being developed. This interest implies a sub-interest with regards to (the contents of) the the system descriptions that are communicated within the community. This interest, in line with [IEEE, 2000], is referred to as the *concern* of a stakeholder:

> *an interest of a stakeholder with regards to the architectural description [*which we will generalise to mean "design description"*] of some system, resulting from the stakeholder's goals, and the present/future role(s) played by the system in relationship to these goals.*
>
> *A concern usually pertains to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders.*

Some examples of concerns are:

- The current situation with regard to the computerized support of a business process.
- The requirements of a specific stakeholder with regard to the desired situation.
- The potential impact of a new system on the activities of a prospective user.

6.3 System development knowledge

The system development community harbours knowledge about the system being developed. To be more precise, objects in the system development community can be regarded as *knowledge carriers* harbouring knowledge pertaining to (their view on) a subdomain within the system being developed (and/or its development process). In this vein, the communication occurring within a system development community essentially aims to create, further, and disseminate this knowledge. The actual knowledge can pertain to the system being developed, as well as the development process itself. In the next section, we will provide a more elaborate discussion on the kinds of knowledge that may (have to) be communicated.

Depending on the concerns of a stakeholder, she will be interested in different knowledge topics related to the system being developed. For example: a financial controller will be interested in an investment perspective on the overall scope of a future system, a designer will be interested in all aspects of the design chain from different perspectives, etc.

6.4 Transformations of knowledge

During the development of a system, the knowledge about the system and its development will evolve. New insights emerge, designs are created, views are shared, opinions are formed, design decisions made, etc. Consequently, the knowledge as it is present in a development community can be seen to evolve through a number of *states*. Knowledge needs to be *introduced* into the community first, either by creating the knowledge internally or importing it from outside of the community. Once the knowledge has been introduced to a community, it can be *shared* among different knowledge carriers. Sharing knowledge between different actors may progress through a number of stages. We distinguish three major stages:

Aware – An actor may become aware of (possible) knowledge by way of the sharing by another actor (possibly from outside the community), or by creating it themselves.

Agreed – When shared, an actor can make up her own mind about the shared knowledge, and decide wether or not to *agree* to the knowledge shared.

Committed – Actors who agree to a specific knowledge topic may decide to actually commit to this knowledge. In other words, they may decide to adopt their future behaviour in accordance to this knowledge.

There is no way to objectively and absolutely determine the level of awareness/agreement/committment of a given set of knowledge carriers. It is in the eyes of the beholder. This "beholder", however, will typically be an actor in the system development community. We can, therefore, safely presume that some actors in the system development community will be able to (and have a reason to) judge the level of sharing of knowledge between sets of actors, and communicate about this.

7 Conclusion

The focus of this paper was on the principles underlying the utility of modelling concepts. We have discussed these issues from three angles: *modelling*, *utility* and *communication*. The primary area of application of the principles presented has been the modelling of enterprise architectures and information system architectures, where the selection of concepts used to model different aspects very much depends on the specific concerns that need to be addressed. Rather than providing 'yet another framework of viewpoints', the aim of this paper was to lay a foundation that enables architects to *reason about* the many frameworks of viewpoints that are already available. We have illustrated our approach by a brief review of the relevant aspects of two existing frameworks for modelling (software intensive) information systems and their architectures. This was followed by a discussion of our perspective on the general role of modelling techniques in system development, and their potential utility.

Currently, we are refining our results in two directions:

- A fundamental understanding of the use of modelling concepts in the art of modelling.
- Refining our perspective on system development as a communication-driven knowledge transformation process.

Both research directions will involve theoretical activities as well as more practical oriented activities based on action research [Blum, 1955, Avison et al., 1999, Baskerville, 1999].

References

[Avison, 1995] Avison, D. (1995). *Information Systems Development: Methodologies, Techniques and Tools*. McGraw-Hill, New York, New York, USA, 2nd edition. ISBN 0077092333

[Avison et al., 1999] Avison, D., Lau, F., Meyers, M., and Nielsen, P. (1999). Action research. *Communications of the ACM*, 42(1):94–97.

[Avison and Wood-Harper, 1991] Avison, D. and Wood-Harper, A. (1991). Information Systems Development Research: An Exploration of Ideas in Practice. *The Computer Journal*, 34(2):98–112.

[Baskerville, 1999] Baskerville, R. (1999). Investigating Information Systems with Action Research. *Communications of the Association for Information Systems*, 2(19).

[Bass et al., 1998] Bass, L., Clements, P., and Kazman, R. (1998). *Software Architecture in Practice*. Addison Wesley, Reading, Massachusetts, USA. ISBN 0-201-19930-0

[Bernus et al., 1998] Bernus, P., Mertins, K., and Schmidt, G., editors (1998). *Handbook on Architectures of Information Systems*. International Handbooks on Information Systems. Springer, Berlin, Germany, EU. ISBN 3-540-64453-9

[Bleeker et al., 2004] Bleeker, A., Proper, H., and Hoppenbrouwers, S. (2004). The role of concept management in system development – a practical and a theoretical perspective. In Gravis, J., Persson, A., and Stirna, J., editors, *Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004)*, pages 73–82, Riga, Latvia, EU. Faculty of Computer Science and Information Technology.

[Blum, 1955] Blum, F. (1955). Action research – a scientific approach? *Philosophy of Science*, 22(1):1–7.

[Boar, 1999] Boar, B. (1999). *Constructing Blueprints for Enterprise IT architectures*. Wiley, New York, New York, USA. ISBN 0-471-29620-1

[Bommel et al., 1991] Bommel, P. v., Hofstede, A. t., and Weide, T. v. d. (1991). Semantics and verification of object-role models. *Information Systems*, 16(5):471–495.

[Booch et al., 1999] Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modelling Language User Guide*. Addison-Wesley, Reading, Massachusetts, USA. ISBN 0-201-57168-4

[Bronts et al., 1995] Bronts, G., Brouwer, S., Martens, C., and Proper, H. (1995). A Unifying Object Role Modelling Approach. *Information Systems*, 20(3):213–235.

[Bubenko, 1986] Bubenko, J. (1986). Information System Methodologies - A Research View. In Olle, T., Sol, H., and Verrijn-Stuart, A., editors, *Information Systems Design Methodologies: Improving the Practice*, pages 289–318. North-Holland/IFIP WG8.1, Amsterdam, The Netherlands, EU.

[Campbell et al., 1996] Campbell, L., Halpin, T., and Proper, H. (1996). Conceptual Schemas with Abstractions – Making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering*, 20(1):39–85.

[Creasy and Proper, 1996] Creasy, P. and Proper, H. (1996). A Generic Model for 3-Dimensional Conceptual Modelling. *Data & Knowledge Engineering*, 20(2):119–162.

[Cruse, 2000] Cruse, A. (2000). *Meaning in Language, an Introduction to Semantics and Pragmatics*. Oxford University Press, Oxford, United Kingdom, EU. ISBN 0-198-70010-5

[Derksen et al., 1996] Derksen, C., Frederiks, P., and Weide, T. v. d. (1996). Paraphrasing as a Technique to Support Object-Oriented Analysis. In Riet, R. v. d., Burg, J., and Vos, A. v. d., editors, *Proceedings of the Second Workshop on Applications of Natural Language to Databases (NLDB'96)*, pages 28–39, Amsterdam, The Netherlands.

[Embley et al., 1992] Embley, D., Kurtz, B., and Woodfield, S. (1992). *Object-Oriented Systems Analysis – A model-driven approach*. Yourdon Press, Englewood Cliffs, New Jersey, USA. ASIN 0136299733

[Falkenberg et al., 1998] Falkenberg, E., Verrijn-Stuart, A., Voss, K., Hesse, W., Lindgreen, P., Nilsson, B., Oei, J., Rolland, C., and Stamper, R. a., editors (1998). *A Framework of Information Systems Concepts*. IFIP WG 8.1 Task Group FRISCO. ISBN 3-901-88201-4

[Finkelstein et al., 1992] Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., and Goedicke, M. (1992). Viewpoints: a framework for integrating multiple perspectives in system de-velopment. *International Journal on Software Engineering and Knowledge Engineering, Special issue on Trends and Research Directions in Software Engineering Environments*, 2(1):31–58.

[Frederiks and Weide, 2004] Frederiks, P. and Weide, T. v. d. (2004). Information modeling: the process and the required competencies of its participants. In F. Meziane, E. M., editor, *9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004)*, volume 3136 of *Lecture Notes in Computer Science*, pages 123–134, Manchester, United Kingdom, EU. Springer Verlag.

[Halpin, 2001] Halpin, T. (2001). *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufman, San Mateo, California, USA. ISBN 1-55860-672-6

[Hofstede and Weide, 1993] Hofstede, A. t. and Weide, T. v. d. (1993). Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100.

[Hoppenbrouwers et al., 1997] Hoppenbrouwers, J., Vos, B. v. d., and Hoppenbrouwers, S. (1997). Nl structures and conceptual modelling: Grammalizing for KISS. *Data & Knowledge Engineering*, 23(1):79–92.

[Hoppenbrouwers, 2003] Hoppenbrouwers, S. (2003). *Freezing Language; Conceptualisation processes in ICT supported organisations*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU. ISBN 9090173188

[IEEE, 2000] IEEE (2000). Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471-2000, IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee. ISBN 0-738-12518-0
http://www.ieee.org

[ISO, 1998] ISO (1998). *Information technology – Open Distributed Processing – Reference model: Overview*. ISO/IEC 10746-1:1998(E).
http://www.iso.org

[Jonkers et al., 2003] Jonkers, H., Hoppenbrouwers, S., Iacob, M.-E., Janssen, W., Lankhorst, M., Leeuwen, D. v., Proper, H., Stam, A., Torre, L. v. d., Veldhuijzen van Zanten, G., Buuren, R. v., Arbab, F., Boer, F. d., Bonsangue, M., Bosma, H., Doest, H. t., Groenewegen, L., and Guillen Scholten, J. (2003). Towards a Language for Coherent Enterprise Architecture Descriptions. In Steen, M. and Bryant, B., editors, *7th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2003)*, pages 28–39, Brisbane, Australia. IEEE Computer Society Press, Los Alamitos, California, USA. ISBN 0769519946

[Jonkers et al., 2004] Jonkers, H., Lankhorst, M., Buuren, R. v., Hoppenbrouwers, S., Bonsangue, M., and Torre, L. v. d. (2004). Concepts for Modeling Enterprise Architectures. *International Journal of Cooperative Information Systems*, 13(3):257–288.

[Kishore et al., 2004] Kishore, R., Zhang, H., and Ramesh, R. (2004). A helix-spindle model for ontological engineering. *Communications of the ACM*, 47(2):69–75.

[Kotonya and Sommerville, 1992] Kotonya, G. and Sommerville, I. (1992). Viewpoints for requirements definition. *IEE/BCS Software Engineering Journal*, 7(6):375–387.

[Kruchten, 1995] Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50.

[Kruchten, 2000] Kruchten, P. (2000). *The Rational Unified Process: An Introduction*. Addison-Wesley, Reading, Massachusetts, USA, 2nd edition. ISBN 0201707101

[Mylopoulos, 1998] Mylopoulos, J. (1998). Techniques and languages for the description of information systems. In Bernus, P., Mertins, K., and Schmidt, G., editors, *Handbook on Architectures of Information Systems*, International Handbooks on Information Systems. Springer, Berlin, Germany, EU. ISBN 3-540-64453-9

[Nijssen and Halpin, 1989] Nijssen, G. and Halpin, T. (1989). *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia. ASIN 0131672630

[Olle et al., 1988a] Olle, T., Hagelstein, J., Macdonald, I., Rolland, C., Sol, H., van Assche, F., and Verrijn-Stuart, A. (1988a). *Information Systems Methodologies: A Framework for Understanding*. Addison-Wesley, Reading, Massachusetts, USA. ISBN 0-201-54443-1

[Olle et al., 1983] Olle, T., Sol, H., and Tully, C., editors (1983). *Information Systems Design Methodologies: A feature analysis*, York, England, EU. North Holland/IFIP WG8.1. ISBN 0-444-86705-8

[Olle et al., 1982] Olle, T., Sol, H., and Verrijn-Stuart, A., editors (1982). *Information Systems Design Methodologies: A Comparative Review*. North-Holland/IFIP WG8.1, Amsterdam, The Netherlands, EU. ISBN 0-444-86407-5

[Olle et al., 1986] Olle, T., Sol, H., and Verrijn-Stuart, A., editors (1986). *Information Systems Design Methodologies: Improving the practice*, Noordwijkerhout, Netherlands, EU. North Holland/IFIP WG8.1.

[Olle et al., 1988b] Olle, T., Sol, H., and Verrijn-Stuart, A., editors (1988b). *Information Systems Design Methodologies: Computerized assistance during the information systems life cycle*. North Holland/IFIP WG8.1, Malham, England, EU. ISBN 0-444-70512-0

[Peirce, 1969a] Peirce, C. (1969a). *Volumes I and II – Principles of Philosophy and Elements of Logic*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA. ISBN 0-674-13800-7

[Peirce, 1969b] Peirce, C. (1969b). *Volumes V and VI – Pragmatism and Pragmaticism and Scientific Metaphysics*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA. ISBN 0-674-13802-3

[Proper and Hoppenbrouwers, 2004] Proper, H. and Hoppenbrouwers, S. (2004). Concept evolution in information system evolution. In Gravis, J., Persson, A., and Stirna, J., editors, *Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004)*, pages 63–72, Riga, Latvia, EU. Faculty of Computer Science and Information Technology.

[Proper and Weide, 1994] Proper, H. and Weide, T. v. d. (1994). EVORM - A Conceptual Modelling Technique for Evolving Application Domains. *Data & Knowledge Engineering*, 12:313–359.

[Putnam, 1975] Putnam, H. (1975). The meaning of meaning. In *Mind, Language, and Reality*, Cambridge, United Kingdom, EU. Cambridge University Press.

[Reeves et al., 1995] Reeves, J., Marashi, M., and Budgen, D. (1995). A software design framework or how to support real designers. *IEE/BCS Software Engineering Journal*, 10(4):141–155.

[TOGAF, 2004] TOGAF (2004). *TOGAF – The Open Group Architectural Framework*. The Open Group.
http://www.togaf.org

[Veldhuijzen van Zanten et al., 2004] Veldhuijzen van Zanten, G., Hoppenbrouwers, S., and Proper, H. (2004). System Development as a Rational Communicative Process. *Journal of Systemics, Cybernetics and Informatics*, 2. To appear.

[Verrijn-Stuart and Olle, 1991] Verrijn-Stuart, A. and Olle, T., editors (1991). *Methods and Associated Tools for the Information Systems Life Cycle*, Maastricht, Netherlands, EU. North Holland/IFIP WG8.1. ISBN 0-444-82074-4

[Wood-Harper et al., 1985] Wood-Harper, A., Antill, L., and Avison, D. (1985). *Information Systems Definition: The Multiview Approach*. Blackwell Scientific Publications, Oxford, United Kingdom, EU. ISBN 0-632-01216-8

[Zachman, 1987] Zachman, J. (1987). A framework for information systems architecture. *IBM Systems Journal*, 26(3).