RC 21266 (94906) 27AUGUST98
Computer Science/Mathematics 10 pages

Research Report

Separating Context and Coordination: Lessons from design wisdom and social theory leading to adaptivity and adaptability through shearing layers

Ian Simmonds
IBM Research Division
T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

David Ing
IBM Advanced Business Institute
Route 9W
Palisades, NY 10964-8001

Research Division
Almaden • T.J. Watson • Tokyo • Zurich • Austin

Separating context and coordination: lessons from design wisdom and social theory leading to adaptivity and adaptability through shearing layers

Ian Simmonds
IBM T J Watson Research Center
30 Saw Mill River Road, Hawthorne
NY 10532, USA
simmonds@us.ibm.com
+1-914-784 7987

David Ing
IBM Advanced Business Institute
Route 9W, Palisades
NY 10964-8001, USA
daviding@ca.ibm.com
+1-416-410 5958

ABSTRACT

The goal of software engineering is to produce effective software systems efficiently. Software systems supporting businesses should effectively support people in doing their work, be it managerial or clerical, highly technical or highly social, exploratory or procedural. At the same time, the practices of software producing organizations should be designed to deliver effective systems on a reasonable schedule and at reasonable cost. Engineering practices determine the nature of resulting software systems, and are themselves typically determined by technology preferences.

This paper presents some findings of an ongoing interdisciplinary investigation into the nature of information systems and the organizations that use and produce them.

In regrounding both information systems and our earlier work on business specifications within a conceptual framework that combines elements of design 'wisdom' and social theory, our underlying assumptions evolve dramatically. We become preoccupied with ideas such as valued incompleteness, federated implementation, just-in-time adaptation by 'users,' and the distinction between technology and its application. At the same time, we lower our expectations in the hunt for reusable domain models, and lessen the distinction between business design and software design.

Key to this regrounding are a distinction between 'context' and 'coordination,' and Stewart Brand's 'shearing layers' model of architectural change.

KEYWORDS

behavioral specification, business design, context, coordination, design wisdom, domain modeling, engineering ethics, interdisciplinary approaches, shearing layers, software design, social theory, systems thinking.

1. A RESPONSE TO ALEXANDER'S CRITICISM

In his keynote speech at OOPSLA '96, architect Christopher Alexander — invited because of the extent to which his work was being cited within the OO design patterns community [19,20] — made the following observation:

> Please forgive me. I'm going to be very directly blunt for a horrible second. But it could be viewed that the technical way in which you [software designers and theorists] look at programming at the moment is almost like 'guns for hire.' In other words, you are the technicians, you know how to make the programs work, 'tell us what to do Daddy, and we'll do it.'

In other words, Alexander was suggesting that today's software engineering theory and practice is — like that of architecture — too oriented towards 'doing the thing right', and at the significant expense of 'doing the right thing'.

So what is the 'right thing?' How do we go about making sure that we 'do it?' This paper attempts to answer these questions for the engineering of business systems.

1.1. Software engineering's 'right thing' is constructing information tools that can be adapted by adaptive purposeful teams as their environment changes

Software developers are often brought into projects of high complexity or with tight delivery dates. As such, they are under a great deal of pressure to 'do the thing right' — to develop or maintain software with high efficiency.

The widely accepted 'right' approach of interviewing and understanding user requirements presumes that the business community itself has a full comprehension of the 'right thing' [28,62]. Unfortunately, since so much human knowledge is tacit [53], there is some probability that users are no more aware of or able to articulate what the 'right thing' is than software engineers.

Thus, we assert that software engineering should preoccupy itself with the search for practices and technologies that allow for *adaptation* of their end product, and in the strongest possible sense of the word. This will involve doing everything conceivable to ensure that software will not get in the way of users as they respond to the ever changing demands of the world. It is about encouraging users to reflect on their own practices, and to take

responsibility for designing the best possible software tools to support their ever improving practices. It is about software technologies and engineering practices that reduce burdens on users as they take on that responsibility.

1.2. Deployment of 'designed things' including software involves intervening in a larger system

All buildings are predictions. All predictions are wrong.

Brand's observation about buildings [10] is also appropriate for organization designs and software.

The full consequences of change to any highly complex and incompletely understood system can *not* be predicted. As complexity theory tells us, even small changes may lead to catastrophe.

Change itself is not the problem. It is the occasionally discontinuous consequences of accumulating small changes (together with unavoidable measurement errors) that causes forecasts and extrapolations to be practically useless. Some recent examples prove this point. If in 1993 someone had predicted that every software developer would need to intimately understand Internet Protocol, they would have probably been laughed off the podium. In 1995, if someone had suggested that C++ would be dethroned by a language proposed for set-top boxes for televisions, a similar response would not have been considered unreasonable.

Yet few change agents — of which software engineers are consciously or unconsciously prime examples — fully acknowledge that each of their *intervention*s may have *systemic*, discontinuous consequences. Every change may lead to an unintended discontinuity (for 'good' or 'bad').

Moreover, Brand reminds us of the need for designing for adaptivity [10]: "An old saw of biology decrees: 'The more adapted an organism to present conditions, the less adaptable it can be to unknown future conditions.'" Of course, the same is true for complex systems in general, and for software and human organizations in particular.

Thus, interventionist approaches focus on ontological and epistemological challenges [28]. The chief ontological challenge is related to how to get a good understanding (or representation) of the target system — is this really how the business works? The corresponding epistemological challenge is to ensure that the method by which we gain the knowledge is 'correct' — how do we know that our interviewees are providing good (open and adaptive) organizational perspectives, rather than a personal bias or a naive or skewed misconception of what is 'right'?

1.3. Rephrasing the business specification / business design distinction as context and coordination

Our survey of design wisdom has led us to reconsider our earlier work on information modeling. In particular, it has led us to reformulate an earlier distinction between 'business specification' and 'business design' as a distinction between 'context' and 'coordination.' The goal of this paper is to document that regrounding based upon our recent interdisciplinary work on understanding the nature of information systems and the organizations that use and produce them.

Design wisdom considers design from the point of view of how 'designed things' are received and understood by their 'users,' and subsequently adapted by them for 'use.' If we believed that these intuitions had reached a sufficient level of maturity we would term them the 'sciences of intervention and design,' but for the moment we refer to them simply as 'design wisdom.' In contrast, there is a scientific theory of human adaptation to change. This is in the social — rather than the natural — sciences, since the ultimate object of study is the human world.

Information modeling [32,33,39] has its origin in a mix of theory, practice and intuition: theory in the form of mathematics and formal methods; practice in the form of considerable experience in the production of large data processing and information systems; and intuitions about the need for, role of, acceptability and accessibility of specifications.

Information modeling is one of a growing number of rigorous approaches to the *behavioral specification* of business and software semantics [36,37,40,41] which are ultimately grounded in mathematical algebras and category theory. It allows for precise specification of behavior (function), whereas software design brings in details of implementation (structure) which is itself determined by current and ephemeral technologies and practices. (Here we use 'function' and 'structure' in the strict technical sense of systems science [1])

A number of higher order concepts and approaches were suggested by experience with the practical application of behavioral specifications [5,35,38,57]. These include: higher order business patterns [42,43]; the role of behavioral specifications in refinement-based approaches to system development [44]; and the management of business rules [45]. In particular, we drew a distinction between business specification, business design, system specification and system design, leading to the extension of the information modeling approach with a notion of 'context' [44]. This paper further refines several of these notions.

However desirable its rigorous mathematical foundations may be, many of the intuitions underlying this earlier work can be better justified in terms of design 'wisdom' and social theory. Moreover, this regrounding better demonstrates the complementarity of information modeling to a wide variety of other information system development approaches — notably systems engineering [59,60] and intervention-conscious approaches [17,28] — allowing us to assemble a much richer, and more comprehensive approach. Indeed we now suspect that we can unify organization design and information system development upon a single conceptual (behavioral) basis.

1.4. Business adaptiveness comes from carefully distinguishing context and coordination

The theme of 'context and coordination,' and the emphasis on design for adaptation, are taken from Steve Haeckel's work on the 'Sense and Respond Organization' [25]:

> Over the last two decades the 'command and control' management system of industrial age firms — characterized by elaborate plans and large central staffs — has been abandoned by many large companies because it proved too slow and inflexible. Companies adopting the sense-and-respond approach must replace command and control with a governance system that provides 'context and coordination' for empowered teams.

With the turmoils of downsizing and then reengineering, 'command and control' has unfortunately degenerated into 'communicate and hope.' The empowerment of individuals has led to a world where they are no longer considered as gears in a mechanism, or organs in an organism, but instead are recognized as purposeful and adaptive. As such, they are looked to to create their own corporate future [3,4].

The design of an adaptive enterprise can be achieved through a dual emphasis on context and coordination. Context sets purpose and bounds of acceptable behavior for an enterprise comprised of individuals, while coordination takes advantage of the human ability to deal with uncertainty and ambiguity. This approach 'shifts the burden' [54] from parts that cannot handle system complexities and ambiguities well — such as business policies handed down from the higher ranks — to other parts of the business system that can adapt and learn — empowered, purposeful groups of people.

Thus the distinction between context and coordination renders explicit and separates the subject of adaptation (context) from the adaptive human agents that bring about change in response to ambiguity and uncertainty (coordination).

2. INDIVIDUALS AND GROUPS ADAPT DESIGNED THINGS ALONG SHEARING LAYERS

Really good design results in designed things — such as buildings — that are 'loved' [10]:

> Age *plus adaptability* is what makes a building come to be loved. The building learns from its occupants, and they learn from it.

The design wisdom that we refer to here is that documented by Stewart Brand in *How Buildings Learn* [10], and can be summarized as follows: designed things must be able to change in qualitatively different ways over qualitatively different time scales.

2.1. Design and time: adaptation is in response to qualitatively different kinds and rates of change

Most designed things are sufficiently durable that they are ultimately adopted and adapted (or rejected and replaced) by several different communities of people. Thus it is not surprising that many similar issues have arisen across radically different classes of design — from cars to space shuttles, houses to factories, startups to multinationals — or that design wisdom can easily be shared across design disciplines.

When Stewart Brand set out to understand the class of complex designed thing called a 'learning organization' he chose, by analogy, to study buildings [10]:

> My approach is to examine buildings as a whole — not just whole in space, but whole in time. Some buildings are designed and managed as a spatial whole, none as a temporal whole. In the absence of theory or standard practice in the matter, we can begin by investigating: What happens anyway in buildings over time?

Among the many fascinating insights in Brand's *How Buildings Learn* [10] is his 'shearing layers' model of architectural change, which draws a distinction between:

> SITE — This is the geographical setting, the urban location, and the legally defined lot, whose boundaries outlast generations of ephemeral buildings. ...
> STRUCTURE — The foundation and load-bearing elements are perilous and expensive to change, so people don't. These *are* the building. Structural life ranges from 30 to 300 years ...
> SKIN — Exterior surfaces now change every 20 years or so, to keep up with fashion or technology, or for wholesale repair. ...
> SERVICES — These are the working guts of a building: communications wiring, electrical wiring, plumbing, sprinkler system, HVAC (heating, ventilation, and air conditioning), and moving parts like elevators and escalators. They wear out or obsolesce every 7 to 15 years. Many buildings are demolished early if their outdated systems are too deeply embedded to replace easily.
> SPACE PLAN — The interior layout, where walls, ceilings, floors, and doors go. Turbulent commercial space can change every 3 years; exceptionally quiet homes might wait 30 years.
> STUFF — Chairs, desks, phones, pictures; kitchen appliances, lamps, hair brushes; all the things that twitch around daily to monthly. Furniture is called *mobilia* in Italian for good reason.

Much of Brand's book consists of illustrations of these qualitatively different kinds and rates of change. One series of time lapse pictures, taken over a period of 6 years, shows changes to the stuff, space plan, services and even detailed structure of part of an old warehouse building as the startup company occupying it grows in size and complexity. Eventually the startup moves out and is replaced with *several* other companies, each with radically different needs yet still able to fit into the same basic structure. The time lapse pictures allow you to see the shearing happening before your very eyes in a manner that goes unnoticed in daily life.

While Brand has already suggested that shearing layers is a concept that can be transferred from buildings to organizations, software engineers should also try to apply it in their own design work. We must ask ourselves, what are the shearing layers of relevance to information systems, within their domains of application, and within their domains of construction?

2.2. Inquiry is a feedback loop through which contexts are adapted as a result of coordination experience

Skilled designers recognize good design when they see it. In his keynote speech at OOPSLA '97, Alan Kay expressed his deep admiration for the design of the US constitution, which was formed complete with counterbalancing institutions to interpret and amend it. It is an expression of shared societal wisdom — societal commitments for governing US society — that flourishes partly because people are free to question everything about it, including the very parts designed to give them that freedom of speech.

The formal design of an inquiry system — as defined by Churchman [13] and, more recently, by Mitroff and Linstone [49] — not only reflects the current knowledge within an organization, but also ongoing learning that allows the organization to continue to adapt. Only when a spirit of inquiry exists does an information system help a business to reach an understanding of its environment.

Haeckel focuses inquiry on context and coordination [25]. Context corresponds to Brand's strong, load-bearing structural layer, while coordination corresponds to stuff and space plan, and is likewise left to the organization's current occupants. Good context allows coordination — on all scales — to occur naturally.

3. SOCIETY ADAPTS BOTH LANGUAGE AND PRACTICES IN SHEARING LAYERS

Human language and practices evolve due to the inevitably different dispositions of people, who are different simply because their trajectories through life cannot be identical. Kittell demonstrates this [47] by tracing the:

> evolution of the office of general receiver of Flanders from 1262 to 1372 and proposes that the emergence of such offices was a crucial factor in the formation of the modern state during the Middle Ages. ... [Y]et careful examination of a wide variety of Flemish financial records reveals that the count of Flanders had no preconceived plan for the creation of a central financial office. Instead, the office was formed and sustained by an internal dynamic in which *ad hoc* actions evolved into administrative routine ...

It is a mistake to think of human practices and language as an integrated whole. They have a richly differentiated structure that is reproduced and evolves — sometimes with discontinuities — in shearing layers and on many different time scales.

3.1. Individual learning occurs on a trajectory through *ad hoc* and *designed* formative contexts

Individual human learning occurs in a manner analogous to Brand's shearing layers in architecture. On the one hand, there are temporary, short term changes analogous to the rearrangement of stuff and space layout. At the same time — and much more subtle and potentially devastating — these slowly occurring and more permanent adaptations accumulate in the deep and buried structures of habit and experience. These latter adaptations are referred to by a wide variety of common sense terms (with various emotive connotations) from 'learning' and 'training' through 'burnout' to 'brainwashing' [30].

As Bourdieu's notion of trajectory suggests [9], potentially all aspects of the contexts through which a person passes may exert a *formative* influence upon them. In particular, as Ciborra notes [14], the software produced by software engineers — along with business procedures, office furniture, and so on — exerts a formative influence on its users, either mental or physical. Thus software should be rigorously designed with both short term and long term formative effects in mind.

It is partly the societal need for careful transmission — from one generation to the next — of accumulated societal wisdom about goodness of disposition and design, that led to the emergence of the reproductive and regulative fields. Also known as education and law, these are enshrined in institutions such as schools and universities, and laws, for example, against the abusive treatment of children.

Yet as we know from design wisdom, any desired formative influences on trajectory are predictions, and predictions are wrong. Thus the institutions (explicitly or implicitly) designed to reproduce societal wisdom must inevitably also foster the continued trial, error and adaptation of evolution.

3.2. Practices shared across teams or industries are slowly evolving and durable, with occasional discontinuities

The reproductive fields of education and law do not exist to produce a homogeneous next generation. Rather, they exist to reproduce [8] a richly differentiated structure of society which consists of many human roles, trades and professions, attitudes and beliefs, and so on. They are built upon the societal wisdom that this diversity is valuable.

In her study of the office of general receiver of Flanders quoted above [47], Kittell illustrates the typical cycle of the *ad hoc* evolution of practices, in which (i) new contexts arise out of necessity leading (ii) to new, improvised practices which (iii) themselves get transmitted through conscious or unconscious mentorship. These practices then (iv) get raised into awareness through the introduction of new language and (v) 'symbolic capital' [7] (in this case, the position of 'general receiver') and (vi) ultimately get incorporated into academic curricula (in this case, the specialized field of governmental accountancy).

Gibson-Jarvie makes a similar point about the City of London [22]:

> In the later seventeenth century [...] coffee houses were opening up, as the new beverage increased in popularity. Lloyd's, Jonathan's, the Jamaica, the Jerusalem, the Antwerp and others established for themselves a special position in the City as the recognized, though not always official, headquarters of many of its budding institutions. [...] Merchants tended to gravitate towards particular houses where members of one or possibly a few trades, or those dealing with a particular locality, would form the major part of the proprietor's clientele.

With his notion of the *cultural arbitrary* [9], Bourdieu points out that this process of evolutionary selection of language for reproduction in successive generations — such as the name Lloyds for London's marine insurance market — is inherently arbitrary. And it is this arbitrariness that led to dramatic differences between long separated cultures, such as that between America and France described in [11]. Yet, it is reflection on this rich variety of differentiated cultures that is the source of much of social theory.

The slow and arbitrary evolution of institutions is roughly comparable to that of Brand's 'site.'

3.3. Language evolves in shearing layers, reflecting the richly differentiated structure of human practices

Particularly important for the fields of information systems and software engineering is the fact that society reproduces dispositions towards the use of language, grammar and vocabulary, reflecting the richly differentiated structure of society.

Bourdieu highlight's this by pointing out the misleading nature of dictionary definitions for words. He points out that [9]:

> concepts have no definitions other than systemic ones, and are designed to be *put to work empirically in systematic fashion.*

In other words, concepts themselves form specialized systems that exist for the purpose of linguistic exchange within specific fields or contexts. Any sharing of 'words' between these concept systems is likely to be an arbitrary coincidence — a remnant of a much earlier state of the evolution of human society in which the differentiations between fields did not exist in their current form.

Since a particular individual may participate in radically different contexts, they may have several radically different uses for each word. Thus, the meaning of a word within a particular context can only ever become fully apparent to those immersed within the context, and then only in its pragmatic relation to other concepts in the context.

3.4. Since language has highly contextual meaning, there is always a risk of misrecognitions

Additionally, much of language is *indexical*, with meaning utterly dependent on the context in which it is used. This is particularly acute in Japanese, as discussed in [6]. The appropriateness of language is always at stake, at the levels both of context (conversation) and field. Moreover, words obsolesce, with fashionable words (like PCTE and C++) being replaced with newer fashionable words (like World Wide Web, Internet and Java), which themselves will obsolesce.

All conversations involve both *recognition* and *misrecognition* due to the very fact that 'the same' dispositions — words, dress, posture, deportment, manners — are (re)used in many contexts and fields. Conversation is as much about determining what the context is (what language to use) and isn't (recovering from misrecognitions) as it is about instrumental action in an identified context.

Moreover, since many conversations are between people who belonging to different fields and with different trajectories, a common ground must be created for linguistic exchanges to take place. Thus, for example, it is unreasonable to assume that even the most apparently uniform of professions, such as tax accounting, will be practiced using the same language even at companies as ostensibly similar as General Motors and Ford, or even between Ford's business units in North America and Europe.

4. REGROUNDING BUSINESS SPECIFICATIONS IN DESIGN WISDOM AND SOCIAL THEORY

Today, the engineering of information systems for use in businesses is usually more about innovation than invention. That is how it should be in a steadily maturing industry.

Technology development is a qualitatively different kind of engineering than technology application. In particular, new technologies should usually emerge in the marketplace, and *not* during application development.

It is against this background that we recommend information modeling [5,38,39,57]. The resulting *business specification* is a suitable basis for the careful yet opportunistic application of structure-bearing media within the business.

4.1. Information systems as part of formative context, along with other media and adaptive humans

As indicated earlier, our emphasis is on design as systemic intervention into whole businesses. Businesses are open systems composed of social and technological parts.

The value of information technologies lies in their ability to enable ("informate" [63]) a 'virtual space and time,' free from many of the bounds of physical space and time. As such, their true value will come when people make innovative use of that virtual space, and both for supportive and formative purposes [14].

Thus we view context as providing both a supportive and a formative function. Moreover, context should be achieved through appropriate means. Far from expecting software to solve all problems, we should pay holistic attention to context. Software — indeed any formative element— should only be incorporated into this holistic design when it is an appropriate provider of a required function.

And we should always remember that the most flexible 'medium' for providing structure for people is other people. This is because a person is an open, and highly differentiated, system of dispositions [9] that can *improvise new structures*, and in an *ad hoc* manner.

This is what we mean by *coordination*, and it is at its best in empowered teams [25].

4.2. How IS relieves both individuals and teams from constraints of space and time

An information system supports humans as they converse and coordinate.

Firstly, technology can remove burdens of memory and familiarity through *context management*, both for individuals and teams. Recall that writing is an ancient

technological aid to memory and context sharing [12]. Context management involves identifying, defining, elaborating, structuring, restructuring, indexing and retrieving content and contexts [16]. It also involves the identification of preexisting contexts, and context merge and split.

One thing that struck us about our earlier work on business patterns [42-5] was how few there seemed to be of the kind we were looking for. We can now explain this small number of patterns and articulate our former intuitions. The earlier business patterns (such as 'information gathering', 'assessment' and 'satisfaction') are all about context creation, and recovery through conversation, and there really are just a few ways of doing that.

Secondly, technology — media — can be used to handle physical separation both in space and time, in presence and in absence [58]. It can allow synchronous interaction across a distance (telecommunication). It can locate and interrupt people wherever they are (track-you-down technologies such as pagers and mobile 'phones). Queuing technologies can support asynchronous communications (e-mail and voice mail). And routing technologies can support communication interception and redirection, ideally following anthropologically sound informal social patterns such as gatekeeper or hub.

Technology can overcome other burdens, but they are beyond the scope of this paper.

Finally — in an attempt to further clarify our opinion as to what IT does and does not offer — we draw a firm distinction between *information* and *knowledge*. For us, while information can be stored in technology, only individual people can have knowledge. Thus, there can be no knowledge technologies, merely socially accepted, knowledge-fostering, applications of structure-bearing media.

4.3. Information technology as structure-bearing media that can also host virtual machines

We use the term *structure-bearing media* to refer to technologies supporting geographically dispersed, possibly asynchronous, ongoing human conversations. They include 'pure' technologies for communication (early telephones) and 'pure' data and information processing, although this purity is long gone and the distinction is obsolete.

We draw a distinction between *structured* and *unstructured uses* of structure-bearing media. This distinction is extremely important, even though a contrast between structured and unstructured uses deliberately emphasizes ends of a spectrum. E-mail and word processors are towards the unstructured end (at which any structure in natural language is largely ignored by the medium), while applications with strong workflow and rigid forms are towards the other.

And we reuse Gelernter's notion of *virtual machine* [21] to introduce forms of information processing beyond the basic ones listed above. These information processing machines are 'virtual' in the sense that they are themselves made out of 'bits' rather than 'atoms.'

Virtual machines provide a spectrum of support for individuals and teams, from the weakly structuring (such as check-as-you-type spell checking in a word processor) to the strongly structured (high-transaction-volume accounting software at the core of a bank). As Gelernter emphasizes, virtual machines can even be used to create yet further virtual machines — this is what programming is all about.

One interesting class of virtual machine is able to recognize structure within an otherwise unstructured medium. This class of virtual machine — which includes that for voice transcription — is of a qualitatively different kind to that which operates upon preestablished structure. In the opposite direction is the idea of 'tangible bits' [31].

4.4. When structure is there, specify it, then apply or build a context-support machine to support it

A structured use of a structure-bearing media requires a commitment to a language of discourse, and this is what a business specification captures [39,41,42,44]. With a structure present, virtual machines can 'process' information in accordance with operations (behavior) that have been specified in terms of that structure (invariant). They regulate the discourse as they enable it.

Languages of discourse — concept systems — enable *conversations* to take place following prescribed rules. Each conversation makes sense in its own terms only. We therefore and define a context as being a seat of a certain logic, in which certain concepts and operations are relevant and make sense *a priori*.

Yet all human conversations mix the highly contextual with the common, and switch from focused to unfocussed (multi-contextual) [21]. Thus software must support both core concepts (structure), and freeform common usage.

A business specification is thus a collection of fragments, each of which contains the *invariant* and *operations* of either a single context or a class of similar contexts [44].

Additionally, for each context there is a need to choose an appropriate *context-support medium*. We formerly termed this business design [44], but now reserve that for the disciplines of the design of commercial organization.

Note that there already exists a rich variety of context-support media, varying from word processors, spreadsheets, blackboards, e-mail, discussion databases, electronic markets, and so on [15,46,51]. We suspect that each of Bourdieu's 'fields' [9] — with its own invariant set of dispositions, including "rules of the game" — has qualitatively different context support needs. Jacobs illustrates this in her comparison of the moral precepts underlying the guardian (political) and commercial (market) syndromes [34]. She shows, for example, how attitudes towards the making of commitments can differ dramatically. In ideal commerce, people 'dissent' (that is, they refuse to make commitments that they are not disposed to keep), whereas ideal guardians use 'deceit' (that is, they believe that the greater good is served by the occasional lie). Thus, we still see considerable scope for innovation in the area of

context-support media, and in the study of how appropriate each medium is for each kind of field or context.

Note that stylized language either works or fails. If it fails then users need to fall back on their pragmatic common sense language for what Habermas calls conversational, discursive and strategic action [23-4,51,56]. Thus a choice of an invariant for a context, albeit partially inherited from involved practices, is a major (and epistemologically problematic) ontological commitment [28].

Thus, in developing adaptive software, we should embed all structure within a rich and fluid language-neutral medium. Further, we should assume that all contexts will need to be adapted, and that all invariants will slowly evolve, and in a shearing layers manner. In other words, software engineers should recognize linguistic phenomena as being just what they are and encourage human linguistic improvisation, innovation and invention.

4.5. Tradeoff and compromise: transfer *versus* coupling in pursuit of inter-context coordination

We start with the intuition that, except for their inflexibility and cost of maintenance, loose federations of legacy systems provide adequate support to many businesses.

Conceptually, we think of a company's information as being added to and stored in a number of distinct 'containers.' Each 'container' stores information pertinent to a single context. (Note that we are saying more than 'one container per class of context.' Rather, we are saying 'one container per context instance.')

Thus inter-context coordination can be understood from the point of view of 'propagation' of newly discovered information throughout an organization's contexts, rather than as the need for a single, integrated repository with centralized and integrated storage of each real or virtual 'thing' [55].

The transfer of information between contexts should be conceived of as a separate operation, just as it was before writing was invented. As such it requires distinct — and specialized — specification. And since contexts frequently have quite different invariants, a transfer operation may well have quite different source and target domains.

The very nature of inter-context exchanges in business often leads to the innovative and explicit design of further contexts, to support these translation and mediation operations. Many businesses have dedicated contexts (departments, procedures) in which specialists act as translators. Indeed, some such contexts have only recently become technologically feasible, such as those built upon data mining technologies. We should also be aware of the slow and steady evolution of fields, in which these interdisciplines become disciplines [48].

Yet an inter-context exchange can only occur between contexts that are aware of each other. In general it is new information that two parts of a company are dealing with the same 'thing,' and thus need to coordinate and converse.

In some industries mutual awareness between contexts is critical, and thus significant investments are made in *mutual awareness systems*. For example, insurance companies and banks seek to prevent misguided or fraudulent behavior (such as bad debts, multiple claims or overinsurance) by securing access — albeit not necessarily immediate — to all coverages and claims involving a particular household across the entire industry. Equally, for practices such as target marketing to be effective, mutual awareness and opportunity pooling are required across a company's lines of business. However, in neither of these cases is there any need for a truly integrated operational view of the enterprise. All that is needed is mutual awareness, and at a reasonable cost.

Mutual awareness technology is ultimately about the sharing of 'thing' identity between contexts [26-7], and the provision of means for searching for contexts that involve a particular 'thing.' It is about having directories of what conversations (contexts) are ongoing, and whom to contact to find out more. It is about "the relationship[s] between the objectives of two or more parties" that may lead to cooperation, competition or conflict [2]. It is about end user-level alarms, notification schemes, 'protocols,' 'locking' or whatever is required when mutual awareness develops.

Of course, at the level of business or application design, an explicit choice may be made to in some way couple contexts at the level of their containers. This can result in more or less coupled or integrated data stores [55] — varying from the loose federation to the integrated— and in some cases to correspondingly 'integrated' data schemas.

However, context integration designs in a hard-to-adapt commitment to the coupling of practices and action. This should only be done if the commitment to coupling also exists at the social level, and the implied restrictions on adaptability are fully understood and committed to by the user organization. Otherwise it builds rigidity into the technology, and breeds resentment at the social level.

4.6. Misrecognition of object-orientation by analysis and design methods

As Alan Kay pointed out in his keynote speech at OOPSLA '97, the term 'object' of 'object orientation' is a bad misnomer for an encapsulating structuring element.

Originally he was inspired more by a biological than a physical metaphor. Highly complex organisms are built out of cells that encapsulate lots of messy 'stuff.' Why not do the same with software? Thus he first considered using the term 'cell,' but later reluctantly accepted the word 'object.'

Unfortunately, many of the conventional practices surrounding object-orientation are built upon this misrecognition of structure as behavior. Domain things do not — in general — behave in and of themselves. They do not, in general take on responsibilities as implied in [62]. Instead, *information about* domain things is *operated upon*, and usually by people. Attributes are ascribed to them *from the outside*, as acknowledged in Fowler's [18] observation- and

measurement-related analysis patterns, which reflect the subjectivity and time variance of 'attributes.'

There is a subtle distinction at work here, largely because in the case of *virtual machines* — such as accounting systems — there may well be *virtual things*, such as accounts, and it may well make sound engineering sense to implement these virtual things as objects (cells). Yet, we assert, many of the things that you want to make virtual — like accounts, plans, contracts, and so on — are themselves simply culturally accepted reifications of culturally reproduced contexts. Indeed, anything that has traditionally been a unit of linguistic exchange or record, written or unwritten is already a reified ('thingified') context.

In general, all support for humans talking about things, be they virtual or not, contexts or not, are highly ephemeral, and should be supported by shearing layers of a suitable ephemerality. Contexts should be recognized as such, even when they already exist reified as 'objects' of the domain. Thus it is important to abandon general purpose system development technologies in favor of distinct techniques and technologies (i) for the design of virtual, context-supporting machines and (ii) for support of humans using language to discuss things, be they virtual and 'real.'

4.7. Misrecognition of the existence of domain models and of change and variation

Another form of misrecognition is that suggested by the 'glossary of terms' and the 'dictionary.' A consequence of this misrecognition is the desire to model industry-wide domain language.

In interviewing subject matter experts in an attempt to construct a domain model for the distribution industry, we were struck by the sheer diversity across the industry: in culture, organization and language.

The language and practices of this industry are highly contextual. Even their contexts are highly contextual. It seems that economic and social forces lead companies to differentiate themselves in terms both of their external, environmental context (market niches) and internal contexts (organization).

Instead, we recommend a search for both generic and industry-specific context-support machines, rather than the modeling of domain language which simply varies too much.

5. SUMMARY AND FUTURE DIRECTIONS

This paper is part of an ongoing joint study between IBM Research and IBM's Advanced Business Institute. We are seeking to further both business and software engineering research through interdisciplinary study.

In this paper, we have outlined how to design information systems as *shearing layers*, in terms of careful application of structure-bearing media in the spectrum from structured to unstructured use. In doing so, we have drawn a strong distinction between technology and the application of technology.

Information technology is expected to provide a spectrum of support from highly structured 'formative context' for novices through to situated meaning [6] and adaptable virtual machines for experts. In general, structured uses are for the novice, while unstructured uses are for experts. Companies must allow novices to be safely productive as they learn, yet allow experts to be creative and adaptive.

Thus we have built upon social theory of education (reproduction) and of practice, and that of Pierre Bourdieu in particular [8,9]. While we are not the first to build on social theory, we do not necessarily "call [...] for systems developers to acquire 'refined skills in organization analysis'" [61]. Rather, we make the methodological observation that if systems developers have undue difficulty producing an information model for a particular context, then they should conclude that the language used in the context is still maturing, and thus that all but slightly structured uses of structure-bearing media are inappropriate.

As for application development, not only is there no need to directly represent domain concepts in code, it is frequently better *not to*. Yet many development methodologies recommend precisely that (for example, in Wirfs-Brock's responsibility driven design [62]). With domain concepts in the code, the code has to be rewritten as the user's language evolves, and today that means dragging in a software engineer.

The only kinds of real world object that we would consider implementing as objects — or preferably, using Alan Kay's original term, as *cells* — are things that reify contexts, including accounts, forms, plans and other previously written forms of record-keeping and communication. And even then, cells should only be used to implement context-support machines. Even for a context-support machine such as the accounting engine specified in [35], records of information about things — real or virtual — and intended for human consumption, should probably not be implemented as cells.

Although the mathematical expression of information modeling [32-3,39] is correct (from the perspective of mathematics and computer science), we prefer to use the terminology of linguistics — homonym, antonym, synonym, connotation, denotation and so on — for discussing issues such as distinction and identity.

We have shown that the idea of requirements divorced from design is misguided. Instead, we should build upon published design wisdom. As we have seen, we need to apply a highly contextual, shearing layers approach — like that of Stewart Brand [10] — to each of language, context determination and context design. We should not unnecessarily or overly couple contexts that may inevitably have differing rates of change and degrees of structure. Any form of rigid and monolithic incorporation of language prescriptions into software prevents adaptive people from adapting software to evolving contexts.

In particular we have suggested how we can 'let emergent things emerge naturally.' Since businesses are becoming

uneasy [50] about technical approaches such as forecasting, so should software engineers. The question for all designers must be: how much do you design into context, and how much do you leave to people?

Equally, as Brand suggests, all complex designed things and their users should be scrutinized through the use of a metaphorical time lapse camera. He points to a number of techniques that allow buildings and their occupancy to be studied over time, including the post-occupancy evaluation (POE) and John Abrams' 'Book' [10]. Presumably software engineering should have counterparts to these.

As designers, we consider it to be an ethical imperative to respond to Alexander's criticism. Behavioral specification and the distinction between context and coordination are our response.

ACKNOWLEDGMENTS

We would particularly like to thank members of the Enterprise Builder and Subject-Oriented Programming teams in IBM Research, and the Sense-and-Respond curriculum team at IBM's Advanced Business Institute — in particular, Bard Bloom, Siobhan Clarke, Steve Haeckel, Bill Harrison, Marianne Kosits, Harold Ossher, Darrell Reimer, Peri Tarr and Mark Wegman — for ongoing discussions on design in general, and organization design and application development in particular. We would also like to thank John Baker, Trevor Hopkins, Haim Kilov, Doug McDavid, Andy Mitchell, Tim Stone, Colin Tully, John Vlissides and May Warren for related conversations over several years, together with the anonymous reviewers for their comments on this paper.

REFERENCES

1 Russell Ackoff, Fred Emery. *On Purposeful Systems.* Intersystems Publications, 1972.

2 Russell Ackoff. *The Art of Problem Solving*, Wiley, 1978.

3 Russell Ackoff. *Creating the Corporate Future*. Wiley, 1981.

4 Russell Ackoff. *The Democratic Corporation: a radical prescription for recreating corporate America and rediscovering success*. Oxford University Press, 1994.

5 Ian F Alexander. *Requirements engineering with Kilov object models*. In [41].

6 Jane M Bachnik, Charles J Quinn editors. *Situated Meaning: inside and outside in Japanese self, society and language*. Princeton University Press, 1994.

7 Pierre Bourdieu. *Language and Symbolic Power*. Harvard University Press, 1990.

8 Pierre Bourdieu, Jean-Claude Passeron. *Reproduction in education, society and culture*. 2nd edition, Sage, 1990.

9 Pierre Bourdieu, Loic JD Wacquant. (1992) *An invitation to reflexive sociology*. University of Chicago Press, 1992.

10 Stewart Brand. *How buildings learn: what happens after they're built*. Viking, 1994.

11 Raymonde Caroll. *Cultural Misunderstandings: The French-American Experience*. Carol Volk translator, University of Chicago Press, 1984.

12 Edward Chiera. *They Wrote on Clay: The Babylonian Tablets speak today*, George G Cameron editor, University of Chicago Press, 1938.

13 C West Churchman. *The Design of Inquiry Systems*. Basic Books, 1971.

14 Claudio U Ciborra. *Teams, Markets and Systems: business innovation and information technology*. Cambridge UP, 1993.

15 Wolfram Conen, Gustaf Neumann editors. *Coordination technology for collaborative applications: organizations, processes and agents*. Lecture Notes in Computer Science 1364, Springer-Verlag, 1998.

16 Basil Doudnikoff. *Information Retrieval*. Auerbach, 1973.

17 Robert L Flood. *Solving Problem Solving: a potent force for effective management*. Wiley, 1995.

18 Martin Fowler. *Analysis Patterns: Reusable Object Models*. Addison-Wesley, 1997.

19 Richard P Gabriel. *Patterns of Software: tales from the software community*. Oxford University Press, 1996.

20 Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

21 David Gelernter. *The Muse in the Machine*. Free Press, 1994.

22 Robert Gibson-Jarvie. *The City of London: a financial and commercial history*. Woodhead-Faulkner, 1979.

23 Jurgen Habermas. *The Theory of Communicative Action, Volume 1: Reason and the rationalization of society*. 1981. English translation, Beacon Press, 1984.

24 Jurgen Habermas. *The Theory of Communicative Action. Volume 2: Lifeworld and system: a critique of functionalist reason.* 1981, English translation, Beacon Press, 1989.

25 Stephan Haeckel. *Adaptive Enterprise Design: The Sense-and-Respond Model*. Planning Review, May/June 1995, Vol.23, No.3.

26 William Harrison, Haim Kilov, Harold Ossher, Ian Simmonds. *From Dynamic Supertypes to Subjects: A Natural Way to Specify and Develop Systems*. IBM Systems Journal, Volume 35, Number 2, 1996, pp. 244-256.

27 William Harrison, Harold Ossher. *Subject-Oriented Programming (A Critique of Pure Objects)*. Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM, Washington, DC, September 1993, pp. 411-428.

28 Rudy A Hirscheim, Heinz-Karl Klein, Kalle Lyytinen. *Information Systems Development and Data Modeling: Conceptual & Philosophical Foundations*. Cambridge University Press, 1995.

29 Patrick Humphreys et al editors. *Implementing Systems for Supporting Management Decisions: Concepts, methods and experiences*. Chapman and Hall, 1996.

30 Aldous Huxley. *Brave New World Revisited*, 1958.

31 Hiroshi Ishii, Brygg Ullmer. *Tangible Bits: Towards seamless interfaces between people, bits and atoms*. CHI '97.

32 ISO/IEC JTC1/SC21/WG7. *Open Distributed Processing — Reference Model: Part 2: Foundations*. (ISO 10746-2 / ITU-T Recommendation X.902, February 1995).

33 ISO/IEC JTC1/SC21. *Information Technology — Open Systems Interconnection — Management Information Systems — Structure of Management Information — Part 7: General Relationship Model*. ISO/IEC 10165-7, 1995.

34 Jane Jacobs. *Systems of Survival*. Vintage, 1992.

35 Haim Kilov, Allan Ash. *How to ask questions: handling complexity in a business specification*. In [41].

36 Haim Kilov, William Harvey editors. *Object-oriented behavioral specifications*. Kluwer Academic Publishers, 1996.

37 Haim Kilov, William Harvey editors. Proceedings of *the 6th OOPSLA Workshop on Behavioral Semantics of Object Oriented Specifications*. San Jose, USA, October 1996.

38 Haim Kilov, Helen Mogill, Ian Simmonds. *Invariants in the Trenches.* Chapter 6 of [36].

39 Haim Kilov, James Ross *Information Modeling: an Object-oriented Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1994.

40 Haim Kilov, Bernhard Rumpe editors. Proceedings of *ECOOP '97 Workshop on Precise Semantics for Object-Oriented Modeling Techniques*. Jyvaskyla, Finland, June 1997, Technical University of Munich, TUM-I9725.

41 Haim Kilov, Bernhard Rumpe, Ian Simmonds editors. Proceedings of *OOPSLA '97 Workshop on Object-Oriented Business Specifications.* Atlanta, Georgia, October 1997, Technical University of Munich, TUM-I9737.

42 Haim Kilov, Ian Simmonds. *Business patterns: reusable abstract constructs for business specification*. In [29], pp. 225-248.

43 Haim Kilov, Ian Simmonds. *Business patterns and viewpoints.* Proceedings of the Workshop on Viewpoints, ACM Symposium on Foundations of Software Engineering, San Francisco, USA, October 1996.

44 Haim Kilov, Ian Simmonds. *How to correctly refine business specifications, and know it*. In [37]. Also IBM Research Report RC 20563 (91041) 9/3/96.

45 Haim Kilov, Ian Simmonds. *Business rules: from business specification to design*. In [40]. Also IBM Research Report RC 20754 (91961) 3/4/97.

46 Mark Klein. *Coordination science: challenges and directions*. pp. 161-176 of [15].

47 Ellen E Kittell. *From Ad Hoc to routine: A case study in medieval bureaucracy.* University of Pennsylvania Press, 1991.

48 Thomas Kuhn. *The Structure of Scientific Revolutions*. Second edition, Chicago University Press, 1970.

49 Ian I Mitroff, Harold A Linstone. *The Unbounded Mind: breaking the chains of traditional business thinking*. Oxford University Press, 1993.

50 Carl Mitcham. *Thinking through Technology: the path between engineering and philosophy*. Chicago UP, 1994.

51 Ojelanki K Ngwenyama, Kalle J Lyytinen. *Groupware environments as action constitutive resources: a social action framework for analyzing groupware technologies.* Computer Support Cooperative Work: the Journal of Collaborative Computing, volume 6, pp. 71-73, Kluwer, 1997.

52 Wanda J Orlikowski, Geoff Walsham, Matthew R Jones, Janice I DeGross editors. *Information technology and changes in organizational work*. Chapman and Hall, 1996.

53 Michael Polanyi. *The Tacit Dimension*. Doubleday, 1966.

54 Peter M Senge. *The Fifth Discipline: the art and practice of the learning organization*. Currency Doubleday, 1990.

55 Dick Schefstrom. *System development environments: contemporary concepts*. pp. 3-95 in Dick Schefstrom, Ger van den Broek editors, *Tool Integration: Environments and Frameworks*, Wiley, 1993.

56 Wes Sharrock, Graham Button. *On the relevance of Habermas's Theory of Communicative Action for CSCW*. Computer Support Cooperative Work: the Journal of Collaborative Computing, volume 6, pp. 369-389, Kluwer, 1997.

57 Mark Shafer. *Experiences related to integrating information modeling with the business requirement definition process at the United Services Automobile Association (USAA) P&C Insurance Company*. in [41].

58 Jim Shedden, Jonas Mekas editors, *Presence and Absence: The Films of Michael Snow*. The Michael Snow Project. Knopf Canada, 1995.

59 Bernhard Thome. *Systems Engineering: Principles and Practice of Computer-Based Systems Engineering*. Wiley, 1993.

60 Colin Tully. *System Development Activity*. Chapter 3 of [59].

61 Chris Westrup. *Transforming organizations through systems analysis: deploying new techniques for organizational analysis in IS development*. pp. 157-176 of [52].

62 Rebecca Wirfs-Brock, Brian Wilkerson, Lauren Wiener. *Designing Object-Oriented Software*. Prentice-Hall, 1990.

63 Shoshana Zuboff. *In the Age of the Smart Machine: the future of work and power*. Basic Books, 1988.