Journal of Network and
Computer Applications 26 (2003) 11–26

Journal of
NETWORK
and
COMPUTER
APPLICATIONS

www.elsevier.com/locate/jnca

Relationships among structural computing and other fields☆

Peter J. Nürnberg[a,*], Monica M.C. Schraefel[b]

[a]*Department of Computer Science, Aalborg Univeristy Esbjerg, Niels Bohrs Vej 8, DK-6700 Esbjerg, Denmark*
[b]*Department of Computer Science, Univeristy of Toronto, Toronto, Ont., Canada M5S 3G4*

Received 1 December 2001; received in revised form 1 April 2002; accepted 1 June 2002

Abstract

We describe the field of structural computing as it relates to a number of other pursuits. By doing this, we hope to accomplish several goals. First, we identify issues that are under- or misrepresented in other research areas—issues that we feel may form the beginnings of a core research agenda for structural computing. Second, we identify points of the structural computing agenda that are as of yet unclear or not yet fully articulated. Third, we point to natural academic allies of the structural computing field—people and ideas who can inform our work, and with whom we may be able to share our ideas.

Keywords: Structural computing; Hypertext; Multiple open systems (MOS); Semantic Web; Structuralism; Postmodernism; Post-structuralism

1. Introduction

Structural computing still feels like a new field. In fact, though, it has been 5 years since the term first appeared in print. There have been three previous workshops on the topic (Reich et al., 2001; Nürnberg, 1999; Anderson and Reich, 2000) each motivating many more papers on the subject. The field still feels young partly because there is as of yet no unifying vision or research agenda to tie together the disparate work done under the label 'structural computing'.

☆ This paper is a modified and extended version of a paper that appeared (Reich et al., 2001).

* Corresponding author. Tel.: +45-7912-7653; fax: +45-7545-3643.
E-mail addresses: pnuern@cs.aue.auc.dk (P.J. Nürnberg), mc@dgp.toronto.edu (M.M.C. Schraefel).

1084-8045/03/$ - see front matter
doi:10.1016/S1084-8045(02)00057-7

In this paper, we continue our recent attempts to define such a research agenda. We do so by comparing structural computing against a number of other pursuits. Specifically, we compare and contrast structural computing from a traditional notion of hypertext, Will et al.'s notion of multiple open services, the recent W3C 'Semantic Web' initiative, and structuralism.

Each of these other pursuits is either related to structural computing by a set of common research problems—it is in their approach to these problems that they differ from structural computing. By comparing and contrasting structural computing with other related areas, especially concentrating on these common problems and differing approaches, we hope to be able to clarify the common thread that runs through the various structural computing work and point out synergies that have so far remained untapped. Also, we use our analysis to help identify areas of the structural computing agenda which are poorly defined, allowing us to concentrate our efforts to address these shortcomings. We hope that this exercise can help bring together structural computing work.

Before we begin with these comparisons, however, we set up a framework for a rough categorization of current structural computing thought in Section 2. Then, in Sections 3–6, we carry out our analysis of structural computing's relationships to the four pursuits named above. In each of these analyses, we begin by describing some relevant similarities and differences between structural computing and the field in question. We then describe some of the most pressing research questions suggested by our analysis of the field in question. Section 7 offers our conclusions on our analysis.

2. A framework for understanding structural computing research

Much work has been done under the moniker structural computing, some of which, at first, may seem only tangentially related. Speaking broadly, we can identify three main areas of research: philosophical; service conceptual; and, architectural. We briefly review these three areas below.

Structural computing began as a *philosophical* critique of hypertext. In Nürnberg et al. (1997), Nürnberg et al. claim that hypertext research problems all share a focus on structure (its creation, manipulation, browsing, etc.), but that the technologies employed by hypertext researchers are fundamentally data-oriented. While this may be viewed as a 'system philosophical' critique, this work has been extended to comparisons with fields such full-fledged philosophical endeavors as structuralism (Nürnberg, 2000) and postmodernism (Nürnberg and Schraefel, 2001). Our analysis of structural computing and structuralism below most closely corresponds with this type of structural computing research.

In a related vein, many claims about a unique *structural conceptualization of services* have been made in structural computing work. Briefly, the claim is that services that are designed and implemented with a special focus on structural abstractions are more convenient and/or efficient to use than 'traditional' services conceptualized within a 'data-first' paradigm. Examples of structural conceptualization of services predate structural computing (e.g. Nürnberg et al., 1996), but neither a rigorous definition of such a conceptualization nor proof of the convenience/efficiency claim have yet been provided in the literature. More recent examples focusing on structural services include work by

Anderson (1999, 2000) and by Wang and Haake (1999). This type of work may be seen as a more concrete version of the philosophical work mentioned above. Our analyses of Wiil et al.'s MOS approach and the Semantic Web work most closely correspond to this type of structural computing research.

Much structural computing work has addressed system *architectural* issues. For example,Wiil and Hicks (2001) describe the architectural implications for supporting an open set of structure servers at the middleware layer of a system architecture—often taken as a basic characteristic of structural computing environments. This approached is based on the widely accepted notion that component-based open hypermedia systems (Nürnberg et al., 1996) provide a reasonable infrastructure for building structural computing environments. Though widely accepted, the reasoning for this assumption has not been formalized in the literature. We believe that this work may be seen as a response to the 'practical necessities' of implementation rather than on any particular theoretical point of the structural computing approach. Nonetheless, such work comprises a significant part of the current structural computing literature. Our analysis of hypertext most closely corresponds to this type of structural computing research.

These three areas of current structural computing research are all based upon the central thesis that focusing on the structural aspects of computing and computing systems, whether at the analysis (philosophical), design (service conceptual), or implementation (architectural) level, yields benefits. The nature of these claimed benefits (and the level of rigor marshaled to their support) vary depending upon the level of the work done.

3. Structural computing and hypertext

Structural computing has most often been closely associated to hypertext[1] by most people, and with good reason. Structural computing has often been interpreted as a generalization of hypertext (Nürnberg et al., 1997), bringing together of various applications of hypertext under one umbrella. This means that problems addressed by hypertext researchers are generally also addressed by structural computing researchers.

As has already been discussed at length previously at the Structural Computing Workshops (Reich et al., 2001; Nürnberg, 1999; Anderson and Reich, 2000), structural computing environments are generally taken to require more flexible and generic infrastructure to support an open set of models than corresponding hypertext environments, which are often specialized to particular tasks. Whereas hypertext environments can (and often do) offer highly specific and optimized infrastructure functionality (e.g. database tables optimized for link resolution), structural computing environments have not yet been able to do so. The optimizations possible in hypertext environments seem to limit the generality of their infrastructure.

Also, hypertext, for wide-ranging reasons, has become popularized. This is not necessarily a bad thing, but it does have consequences. Hypertext is a term that has been appropriated (or misappropriated) by many different people with many different agendas. This means the term itself has become increasingly blurry, with the term 'hypertext' being

[1] We understand the terms '''hypertext' and 'hypermedia''' to be interchangeable.

used to describe everything from multimedia presentations to help system interfaces to the World Wide Web to postmodern fiction. Its popular usage may not correspond to any traditional usages. This is most clear on the critical theory front, in which hypertext has been appropriated by critics seeking an embodiment of post-structuralist critical thought. Hypertext as postmodern technology has been a siren song to many theorists, even though recent discussions by critics have questioned this idea. Nonetheless, hypertext, for better or worse, is seen by many as affiliated with a set of particular theoretical programs. Structural computing, conversely, is not yet popularized. We still have our pick of allies. (We visit some of these potential allies in sections below.)

3.1. What is the generic structural abstraction?

This question was posed very early on in the structural computing line of work, but has not since been addressed. If structural computing infrastructure should support an open set of structural abstractions, what abstraction(s) should be made available by the infrastructure? A related issue concerns the relationship between data and structure. Is structure a 'superclass' of data, or just a peer? Structural computing has asserted the former, but how can we show this to be true (or untrue)? Is this even demonstrable? We examine some of the philosophical implications of generic abstractions (as fundamental) below.

Currently, the generic structural abstraction, or 'structural atom', in both Callimachus (Tzagarakis et al., 1999) and Construct (Wiil and Nürnberg, 1999) is a highly generic abstraction with 'built-in' structuring capabilities. That is, these atoms are able to be related to one another. In Construct, for example, atoms have various types of attribute-value set pairs. One type of pair is the 'relation' type, in which values in the value sets are identifiers of other atoms. Other types allow for expressing traditional types of attribute values, in which values take on text, binary, or other specialized formats.

To what degree are these abstractions even able to be called 'structure?' In general, the argumentation is as follows. When a new problem domain is encountered, models are built to represent both the data and structure found in these domains. Middleware servers are generated to export these models and their abstractions to clients. The abstractions (both data and structure) exported by these servers are then represented by structural atoms in the backend. In this way, structure (as identified at the middleware layer) is given first-class status in the backend. Data can be seen as a degenerate case of structure in which the structuring is as of yet unidentified. (That is, the atoms that represent data abstractions do not yet take advantage of the relation type of attribute-value set pair, although nothing prevents these atoms from doing so at a later time.)

The structural atom abstraction of the backend in these systems, however, may be nothing more than 'structured data'—data (just as in hypertext systems) that always *may* stand in relation to other (structured) data. Is this what was meant by raising structure to first-class status? Or did we simply inherit a data-centric systems view from our hypertext colleagues? It seems that although structure *at the middleware layer* is now first-class, structure at the backend is not. That is, that which relates two atoms (an instance of the 'relation' attribute-value set type) is still relegated to second-class status, unequal to the atoms it relates. Perhaps this is sufficient. Or, perhaps what is needed here is the continuing

'push' of structure-awareness farther 'down' into the system, all the way to the backend itself (Nürnberg et al., 1996).

3.2. How can infrastructure performance be optimized with generic abstractions?

If structural computing infrastructures support generic abstractions, how can they achieve efficiencies similar to the highly optimized infrastructures of modern hypertext environments? At first, this seems to be impossible—optimizations based on semantics of structural elements (something of which the infrastructure is not normally aware) would seem to violate the layering of typical structural computing architectures? Are there other approaches? Wiil and Hicks have recently begun to discuss alternatives to layered architectures (Wiil and Hicks, 2001). Can other such architectures help structural computing architects to deliver efficient infrastructure services?

Another interesting direction is the idea of *dynamic optimizations*, first explored in the hypertext area in the Coconut project (Nürnberg and Grønbæk, 1999). In this system, designs called for a special-purpose API to the backend to allow 'parallel' storage of abstractions both in generic and specialized forms (in separate database tables, for example). Specialized tables would be used for quick lookup. Attribute-value set pairs associated with an atom that fell outside the templates used in specialized tables were attached to the generic form of an atom. Transaction control was to be used to manage separate copies in multiple table of the identical atom. This functionality was never implemented, however, so the validity of this direction is still speculative.

Modern component frameworks such as CORBA (OMG, 2001) and Java Beans (DeMichiel et al., 2001) provide another possibility for mitigating performance penalties associated with overly general backend abstractions. These systems both allow conceptual or logical components to reside in the same process space. A similar approach for improving component performance was used in the SP3 (Leggett and Schnase, 1994) and HOSS (Nürnberg et al., 1996) systems. Thus, even if structural computing environments require additional conceptual architectural layers by specifying highly generic backend abstractions, the performance penalties associated therewith may be able to be minimized by clever 'layer collapsing' implementations.

3.3. How can interoperability be supported?

Interoperability has meanings specific to structural computing environments. Since open sets of structural abstractions may exist within such environments, 'mapping strategies' for interoperability that may be appropriate in more limited hypertext environments may fail in structural computing environments. Interoperability is not simply a desirable trait for structural computing environments—it is a necessary characteristic. By this, we mean not only inter-system interoperability (supporting a plug-and-play metaphor, whereby services from one environment can interact with services and applications from another), but also intra-system interoperability (supporting a way in which structures and behaviors from one service can be used by another service). Anderson has presented intriguing work in this direction (Anderson, 2000). Kyriakopoulou et al. have recently begun to discuss specialized structure-oriented dynamic service

discovery (Kyriakopoulou et al., 2001). Can this be a starting point for general structural interoperability? Have we even begun to think about interoperating behaviors?

Inter-system interoperability has an established history in the hypertext field. An excellent example of this is the Open Hypermedia Protocol (OHP) work (Reich et al., 1999), carried out by the Open Hypermedia Systems Working Group. The OHP work, despite its name, focused on defining interfaces to various middleware services. Interface definitions take the form of IDL. A number of such interfaces were developed, with several systems implementing the most common of these interfaces. The Construct structural computing environment has implemented some of these OHP interfaces. However, thus far, structural computing environment interoperability has not yet been demonstrated.

Intra-system interoperability has been addressed in structural computing and hypertext settings. The first working example of intra-system interoperability between different structure types (middleware services) in a structural computing environment was the CAOS system (Reinert et al., 1999), in which structures generated by either the spatial structure or the navigational structure services could be 'reinterpreted' by the other. More recently, the FOHM work (Millard et al., 2000) has tried to formalize intra-system interoperability, albeit for a limited set of services. Although initially promising, it is unclear how the FOHM work can be generalized, and how it can be applied to structural computing environments, with their more generic backend abstractions.

4. Structural computing and multiple open services

Wiil (1999) posed a challenge to the structural computing community at the very first structural computing workshop—is structural computing simply a special case of the multiple open service approach to service provision? Discussions at that workshop did not resolve this challenge. Here, we look more closely at the relationship between these endeavors.

The primary determining architectural characteristic of structural computing environments versus traditional hypertext environments was the radical middleware layer openness—an architectural axiom that structural abstractions comprise an unbounded set, and so, therefore, do structural services. Wiil claimed that multiple open services generalizes structural computing. In what way can this be understood?

One possible interpretation is that multiple open services stresses openness at the infrastructure layer as well as the middleware (and client) layer(s). This objection seems reasonable, at least inasmuch as openness at this level is in fact not stressed in structural computing work (although it clearly is a part of existing structural computing systems, such as Callimachus (Tzagarakis et al., 1999) or Construct (Wiil and Nürnberg, 1999). Indeed, in Wiil et al. (2001), Construct is used to illustrate the multiple open services concept.) Nonetheless, the multiple open services approach does make more explicit its radical infrastructure openness. Also, there is also explicit mention of integration of development tools into the environments they help develop—a 'bootstrapping' focus not explicitly present in structural computing work to this point.

However, we believe the intended objection goes to the heart of seeing services that are unspecified (as they are in the multiple open services conceptualization) as a generalization

of services that are structural. Stated in this way, the point seems to be axiomatic. This, we believe, is the crux of the claim made by Wiil.

The error in this conceptualization, however, is in this very seemingly axiomatic formulation. Structural services are not a subset of services in general, as, in principle, they are not concrete instances of service per se, but conceptualizations of services in general. The structural computing theoretical argument is that *any* service is better seen (and treated) as a structural service regardless of implementation. Thus, any service admitted by a multiple open services environment (or any service in general) is also admitted by a structural computing environment. It is this reconceptualization that is at the heart of structural computing.

4.1. Are there examples of truly non-structural services?

The gauntlet has been thrown down. The claim that all services, regardless of implementation, are better viewed as structural services, is now on the table. Is this true? Or, are there in fact services that suffer from a structural recasting? Disproving the hypothesis should be straightforward enough, provided we can define what it means to 'reconceptualize' a service in structural terms. This issue is intimately related to the nature of the 'generic' structural abstraction discussed above (see Section 3).

What does it mean to recast a service 'in structural terms?' As above, current structural computing environments view *structure* as a middleware service issue, whereas *structured data* is viewed as a backend issue. From a backend perspective, this recasting seems straightforward—simply allow that any abstraction instance potentially be able to stand in relation to other instances. But what does this recasting mean in the middleware layer? Or, in a related vein, what does it mean to recast a service in structural terms if the backend itself is truly structure-oriented?

To date, the community has seen this as an abstraction 'status' issue. Relationships among abstractions are raised to first-class status by allowing them to participate in other relations. If viewed in this way, recasting services as 'structural' seems only to imply modification by addition—given any service, its data-oriented abstractions may not need to change, even if its structural abstractions are raised to first-class status. This would imply a fairly non-radical change. Perhaps the claim is less controversial than it sounds.

The more radical interpretation of the claim, however, is that a service's structural abstractions should become even *more* important than its data abstractions. What could this mean? It seems clear that such a radical reading has yet to be practiced by the field's practitioners. This may suggest an interesting avenue of inquiry for the field.

4.2. How do we leverage the multiple open service insights?

How can we take (co-opt?) the work at the lowest levels of architectures stressed in the multiple open service work? Although not discussed in Wiil et al. (2001), one could argue that a multiple open service environment is more naturally modeled with a non-layered architecture. (In fact, the notion of layer is rather muddled in the current multiple open service literature—this may point to the fact that it may be largely unnecessary or even harmful.) If multiple open service work does break the layer model, can adopting

the multiple open service approach to infrastructure and development tools, with its architectural implications, inform our previously mentioned inquiry into infrastructure optimization?

5. Structural computing and the Semantic Web

A semantic web (as described by Nelson, and recapped in the W3C's use of his Scientific America article (Berners-Lee et al., 2001) proposes to let web pages continue to be web pages—that is to appear semi-structured in database terms, but to be at least tagged structurally behind the scenes. Such tagging will allow dynamic structures to be built across a heterogeneous web space in order to address complex user queries, such as 'where did David get his PhD.' The glue for such queries across multiple, heterogeneous sites consists of ontologies. An example of an ontological system for a semantic web can be found at Miles-Board et al. (2001).

Much of the current Semantic Web work has been focused on the information retrieval aspects of the problem: how to effectively tag information with metadata so that it can be processed by machines. To that end, there have been papers on the development of metadata markup languages such as RDF (Swick, 1999) or extending RDF (Broekstra et al., 2001) to better support higher-level knowledge representation. Other work, such as van Ossenbruggen et al. (2001), has focused on the particular needs of metadata for multimedia. Other research, for instance in Maedche and Staab (2001), has considered building tools for designing the ontologies themselves. The emphasis in this work has been on what may be characterized as a 'data first' approach where defining properties to represent the data for machine access is of primary importance. Structure is relegated to the more or less implicit predicate of a metadata relation. For instance, RDF supports the concept of representing the seemingly inherent properties of the data as predicates 'published by' or 'effective on;' this document is published by that group; this sale takes effect on that date. More complex relations (ontological ones) of information objects are reflected in object class hierarchies expressed in such languages as DAML and OIL (van Harmelen and Horrocks, 2001) which extend RDF.

Part of structural computing's emphasis is to treat structure as a first class entity. Based on the Semantic Web's emphasis on data and metadata, it may be possible to construct a relation between the Semantic Web and structural computing as a special case of the relationship between information retrieval and hypertext. Information retrieval, however, is geared to automatic processing of information, whereas hypertext is geared to structures authored and browsed by people. Structural computing was born of hypertext work, whereas the Semantic Web, with much of its focus on ontological support, may be seen as more closely akin to information retrieval than hypertext. This observation has not gone unnoticed in parts of the hypertext community. As Goble et al. note in their Open Hypermedia work on the COHSE system for improved, Web-based document linking:

> In the recent enthusiasm to move the Web to one that has machine understandable semantics for automated processing we are in danger of forgetting that it has been successful not only as a machine readable infrastructure but also as one that is browsed

> by humans. In other words, the Semantic Web is still a web, a collection of linked nodes. Navigation of links is currently, and will remain for humans if not machines, a key mechanism for exploring the space. This should not be lost. The Semantic Web is viewed by many as a knowledge base, a database or an indexed and searchable document collection…we view it as a hypertext (Goble et al., 2001).

In the COHSE work, the emphasis is to provide a suite of services to use metadata annotations to create hypertexts. As the authors note, this is distinct from an information retrieval approach, which looks for a pre-existing resource to suit a specific query. That said, the emphasis of COHSE is more on creating hypertexts by dynamically linking pre-existing documents (no mean feat to resolve appropriate link/document selection), rather than on creating new composites of appropriate documents. It has been noted that any system reflects the artifacts of the assumptions that go into its design (Carroll and Rosson, 1992). One of the assumptions of a data-first approach to hypertext, even as reflected in the COHSE project which is more oriented to integrated services to support user exploration, rather than to machine-oriented information retrieval, is the primacy of the data 'as it is.' Here, making a hypertext is seen as the non-trivial operation of adding links to a preexisting document. Operations on the document itself—of making the 'found' data available to different processes are not particularly considered. This is not a criticism of the Semantic Web work, or of the COHSE project in particular; it is an observation that current Semantic Web research, which seems a natural place to consider structural architectures, is strongly informed by a bias towards data as primary and structure as implicit service.

Structural computing itself, with its roots in hypertext has gone farther towards foregrounding structure in discussing general structure. While we have often understood this to mean 'general hypertext structure,' there is no reason why we should limit ourselves in this way. Nürnberg et al. (1996) point to a variety of non-hypertext structural possibilities (e.g. conceptualizing 'make' or 'cron' as structure services, building structure-aware kernels, etc.) Structural computing may most properly be viewed as being just as concerned with automatic, machine-processable structures (e.g. the ontologies of semantic webs) as with hypertext structures, but it is also interested in considering other structural possibilities that can be accessed and orchestrated together, as proposed by, for instance the Callimachus project on developing service templates for component-based open hypermedia systems (Tzagarakis et al., 2003).

5.1. Can structural computing environments support semantic webs, information retrieval and/or other 'machine-processable' structures?

Taking for instance the services proposed in COHSE or the RDF-based annotation services proposed in Kahan and Koivunen (2001) and the primitives and templates defined in Callimachus, a Semantic Web as link discovery or annotation service may be seen as simply a set of potential middleware services within structural computing environments. Similarly, the more information retrieval or databases oriented processes may also be supported as services. There appear to be recent developments in these directions. In earlier work, Wiil et al. (2001) describe using Construct to support traditional metadata

services. We can build upon these early ideas. These issues are probably tightly bound to the efficiency issues described above (see Section 3).

5.2. How should we repackage our work as more general infrastructure?

The Semantic Web, with its emphasis on offering services for accessing information provides an opportunity to promote the advantages of structural computing both as a set of design principles for service descriptions and for service reuse in an Open Hypermedia environment. We have been ineffective in selling the structural computing work outside of the hypertext and digital libraries communities. How can we begin to disseminate our work outside of these limited areas? Should we as a workshop community form a plan for which communities to target next? Anderson (1999) has discussed exporting structural computing technologies to the software engineering communities extensively. Wang and Haake (1999) have described connections between structural computing and computer supported collaborative work. How can we leverage these footholds? Also, how can we draw in people from these fields into structural computing to help inform our work?

6. Structural computing and structuralism

In previous work (Nürnberg (2000)), the potential relation of structuralism to structural computing was touched upon. The question was raised as to whether or not structural computing should be reframed as structuralist computing. We would like to say 'No' for the time being, due to the rationalization for the structuralist project across linguistics, psychology, philosophy, narratology or any other discipline that claims a structuralist approach. That is, structuralism is in pursuit, as mentioned in Piaget (1996), of 'laws' or foundational structures through which understanding can be known and discovered. These laws take the form of discovered structures or 'transformations…inherent in the object' (Lévi-Strauss, 1963) which can refer to an innate or even genetic source through which we can understand ourselves. Even Chomsky's transformational grammars refer back to a Cartesian 'innate rationality.' As such, structuralism seems, to put it over-simply, rather deterministic. This innate determinism is something with which the existentialists, contemporaries of high structuralists, took great issue.[2]

We, too, take issue with this determinism, if for different reasons than the existentialists. It does not seem that the goal of structural computing (at least at this point, and perhaps at no point) is to attempt to claim that we can, through the experimental Structure First model of structural computing, and its investigation of the way we build computational architectures, determine an innate understanding of Who We Are, fundamentally, as a species (or how we are, Structuralism might suggest, programmed to be). Let us suppose, however, that the models we choose do tell us something about ourselves here and now, rather than ultimately. Perhaps what they tell us in sifting through

[2] The reader is invited to look at the history of philosophy with respect to the ongoing antagonism and animosity between Levi-Strauss and Lacan's Structuralism and Sartre and de Bauevoir's Existentialism. For a cogent discussion and critique of these relations, see Brodribb (1992) and Moi (1994).

structural computing architectures is nothing new, but an affirmation of the known. Still, this is a question worth considering: what is being reconfirmed?

6.1. What was the question?

Structural computing's willingness to foreground the relation of structure to data creates a place to discuss the philosophy of computation or the political science of computational models. (We use the term 'science' with respect to its Latin origins as 'knowledge,' not as objectivity or absolutism.) Some may suggest that these discussions have taken place in hypertext for some time. We would like to suggest that they have, but have been geared to a particular audience and from a particular audience—namely, literary critics who have found the pleasure of the (hyper)text (Barthes, 1975) in a mainly post-modern way, and, understandably, at the surface of the media, in its textual presentation and interaction rhetoric. The Structure and Data pairing, however, also raise distinct questions at the architectural and system level. These questions are interesting to consider from an engineering context and are raised by structural computing's attempts to shift the focus of computing from Data to Structure. What are the implications for computing (and beyond) from such a shift? Why are they worth considering?

Marx proposed a consideration of work and work relations to capital as the site of profound social investigation (Marx, 1990) (see also Tucker (1978), e.g. p. 443–65)[3] We propose that the philosophical consideration of structural computing may also act as a site of, if not profound social investigation, at least a more foregrounded investigation into our own practices, assumptions and their consequences, once we consider that engineering is not innocent (not distinct from other cultural practices), nor our approaches absolute. What are the biases and beliefs that both inform our technology and are created by that technology? Ursala Franklin, a renowned metallurgist, defined technology as 'Practice,' as a 'Way of Doing Things' (Franklin, 1999); and, claimed that we are identified as belonging or not to a particular group based on our technology, on how we do things.

Structural computing suggests a new practice with respect to the relation of structure and data, and in Franklin's sense, therefore, proposes a new technology. We are fortunate that while in the process of creating that technology, we are reflecting on the implications of its definition. The questions we can ask, therefore, concern why we think that the shift to Structure First will be 'better' than the Data First approach. We do not have proof yet that such is the case, but we have experiments we wish to try to demonstrate the value of this approach at least in certain contexts. However, we should not dismiss the potential profundity that such a shift may imply at the more general levels of our technology, our practice, in pursuing such an approach. We do not pretend to have yet worked out all the implications of this shift, but we would like to mention one in particular for further consideration: the relation of structural computing to Structuralism. We will then get more

[3] Marx is also frequently considered part of the Structuralist way, since his work is often seen to remove human action (and history) from impact on the system. This determinism, however, is a site of regular controversy among branches of Marxist thought, such as Marxist Feminism and the Frankfurt school. A valuable overview of Marxism is available in Bottomore (1991).

practical, in terms of how this approach might be applied for its functional efficacy to be evaluated.

6.2. What is implicit in the Structure/Data Pair?

So let us put the question again, perhaps in a more Deconstructionist[4] way: what happens when we challenge the primacy of Data over Structure by inverting the pair to say structure should lead, or at least become on the same level as data? That data does not determine structure, but structure data? Are we making structure data-like, or data a lower class form of structure? Or something else? And if any of these, therefore what are the costs to the model of computer science that foregrounds Data rather than Structure? A more Marxist-informed question about the shift may ask, whose interests are being served by redefining these relations?

In the interim, one practical, lower level effect may be to provide a rationale to eliminate fixed markup in a hypertext. In a sense, the Semantic Web is heading in this direction in terms of data-based queries across heterogeneous sites, where markup's role is semantic rather than presentational. Structural computing suggests that even semantic markup needs to be considered as contextual, to be determined by the structure to represent that data at any time.

The shift from Data to Structure problematizes models of representation and perhaps notions of originality, so germane to discussions of intellectual property in the digital domain. That is, in the Structure First model, does data become seen as a kind of neutral receptor of multiple structures (and therefore, sampling and recontextualization are where the IP is, rather than in any single contextualization of the data), rather than something that is simply inseparable from structure? If the former, it would seem then that data becomes almost atomistic, pre-existing structure, and therefore primary again (a sort of structural linguistics-like position). If the latter, then at least we recognize that data and structure are toujours deja implicated in each other. As such, if data becomes the 'simpler case' of dealing with structure in a computational model, the corollary of this means that structure is always present. Data is not structure neutral, not atomistic. This constant implied relation of structure and data is as close as structural computing may come to a Structuralist perspective of the system informing relations within the system. We resist the next step which would suggest that the structures we discover or models we build are fundamental. After all, the relation of structure and data is itself a constructed relation and we are just as interested in the implications of that construction as the effects of its inversion.

[4] It seems relevant if beyond the scope of the paper to dwell on the differences between post-structuralism and deconstruction. Again, to oversimplify, post-structuralism, as embodied in Barthes *S/Z* (Barthes, 1974) foregrounds how one can continue to define not one but multiple structures from any proposed ultimate structure. Deconstruction, on the other hand, as reflected in Derrida's *Of Grammatology* (Derrida, 1976), or more briefly in his article 'Structure, Sign, Play,' (Derrida, 1978) is focused on oppositions proposed by any approach—Good/Evil; Speech/Writing—and uses those to undo the assumptions which hold one part of the pair in primacy over the other. In our case this would be Data/Structure.

7. Conclusions

In the above sections, we have attempted to define structural computing by analyzing four closely related areas. We have postulated where we can look for near-future application challenges. Structural computing has echoes in other areas of computing and philosophy, only some of which are touched upon above. Part of our goal of course should be not to reinvent the wheel. Therefore, we need to be interdisciplinary in our approach to develop structural computing as school of thought and as technology. Part of our mandate must be to reach out beyond our group to research (and researchers) in these other domains to critique, inform and enrich our practice. We deliberately seek consultation and collaboration with these other areas. By this, we make our practice potentially more deliberately socially constructed.

Similarly, we are aware that at this point we seem to have more questions than answers, but we believe that the questions are valuable and have the potential to create both effective models for improved computational delivery of information and potentially new ways of interpreting computational practice.

References

Anderson KM, Reich S (Editors). Proceedings of the Second Workshop on Structural Computing (SC2), Lecture Notes in Computer Science, vol. 1903, Springer, Berlin, 2000.

Anderson KM. Software engineering requirements for structural computing. In: Núrnberg PJ, editor. Proceedings of the Workshop on Structural Computing (SC1), no. AUE-CS-99-04 in AUE-CS technical reports, Aalborg University Esbjerg. Esbjerg, Denmark: Aalborg University Esbjerg; 1999. p. 22–5. URL http://cs.aue.auc.dk/publications/

Anderson KM. Structural computing requirements for the transformation of structures and behaviors, in: K.M. Anderson, S. Reich (Editors), Proceedings of the Second Workshop on Structural Computing (SC2), Lecture Notes in Computer Science, vol. 1903, Springer, Berlin, 2000, pp. 140–6.

Barthes R. S/Z, New York: Hill and Wang. 1974. Richard Miller (trans.).

Barthes R. The pleasure of the text, Hill and Wang, New York, 1975.

Berners-Lee T, Hendler J, Lassila O. The Semantic Web, Scientific American, 2001. URL http://www.sciam.com/2001/0501issue/0501berners-lee.html

Bottomore T (Editors). A dictionary of Marxist thought, 2nd ed., Blackwell, Cambridge, 1991.

Brodribb S. Nothing mat(t)ers: a feminist critique of postmodernism, Spinifex Press, North Melbourne, 1992.

Broekstra J, Klein M, Decker S, Fensel D, van Harmelen F, Horrocks I. Enabling knowledge representation on the Web by extending RDF schema. Proceedings of the Tenth International World Wide Web Conference. International World Wide Web Conference Committee; 2001. p. 467–78.

Carroll JM, Rosson MB. Getting around the task-artifact cycle: how to make claims and design by scenario, ACM Trans Inform Syst 10 (2) (1992) 181–212.

DeMichiel LG, Úmit Yalçinalp L, Krishnan S. Enterprise JavaBeans specification version 2.0. Tech. rep., Sun Microsystems; August 2001.

Derrida J. Of grammatology, Johns Hopkins University Press, Baltimore, 1976, Gayatri Spivak (trans.).

Derrida J. Structure sign and play. Writing and difference, University of Chicago Press, Chicago, 1978.

Franklin U. The real world of technology, CBC Massey Lectures Series, revised Edition, Anasasi Press, Toronto, 1999.

Goble C, Bechhofer S, De Roure D, Hall W. Conceptual open hypermedia = the Semantic Web? Semantic Web Workshop 2001, CEUR Workshop Proceedings, Hong Kong, China; 2001. URL http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-40/

Kahan J, Koivunen MR. Annotea: an open RDF infrastructure for shared web annotations. Proceedings of the Tenth International World Wide Web Conference. Hong Kong, China: International World Wide Web Conference Committee; 2001. p. 623–32.

Kyriakopoulou M, Avramidis D, Vaitis M, Tzagarakis M, Christodoulakis D. Broadening structural computing towards hypermedia development, in: M. Tzagarakis, S. Reich (Editors), Proceedings of the Third Workshop on Structural Computing (SC3), Lecture Notes in Computer Science, Springer, Berlin, 2001, pp. 131–140.

Leggett JJ, Schnase JL. Viewing Dexter with open eyes, Commun ACM 37 (2) (1994) 76–86.

Lévi-Strauss C. Structural anthropology, Basic Books, London, 1963.

Maedche A, Staab S. Learning ontologies for the Semantic Web. Semantic Web Workshop 2001, CEUR Workshop Proceedings, Hong Kong, China; 2001. URL http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-40/.

Marx K. Das Kapital, A Critique of Political Economy, vol. 1, Penguin/New Left Review, London, 1990, Ben Fowkes (trans.).

Miles-Board T, Kampa S, Carr L, Hall W. Hypertext in the Semantic Web. Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia (HT '01). New York: ACM Press; 2001. p. 237.

Millard DE, Moreau L, Davis HC, Reich S. FOHM: a fundamental open hypertext model for investigating interoperability between hypertext domains. Proceedings of the Eleventh ACM Conference on Hypertext and Hypermedia (HT '00). New York: ACM Press; 2000. p. 93–102.

Moi T. Simone de Beauvoir: the making of an intellectual woman, Blackwell, Cambridge, 1994.

Nürnberg PJ, editor. Proceedings of the Workshop on Structural Computing (SC1), no. AUE-CS-99-04, in AUE-CS technical reports, Aalborg University Esbjerg. Esbjerg, Denmark: Aalborg University Esbjerg; 1999. URL http://cs.aue.auc.dk/publications/

Nürnberg PJ, Grønbæk K. A component-based open hypermedia approach to integrating structure services. New Review of Hypermedia and Multimedia 5, 1999.

Nürnberg PJ, Leggett JJ, Schneider ER. As we should have thought. Proceedings of the Eighth ACM Conference on Hypertext and Hypermedia (HT '97). New York: ACM Press; 1997. p. 96–101.

Nürnberg PJ, Leggett JJ, Schneider ER. HOSS: a new paradigm for computing. Proceedings of the Seventh ACM Conference on Hypertext and Hypermedia (HT '96). New York: ACM Press; 1996. p. 194–202.

Nürnberg PJ. Repositioning structural computing, in: K.M. Anderson, S. Reich (Eds.), Proceedings of the Second Workshop on Structural Computing (SC2), Lecture Notes in Computer Science, vol. 1903, Springer, Berlin, 2000, pp. 179–84.

Nürnberg PJ, Schraefel M. Structural computing and it relationships to other fields, in: S. Reich, M. Tzagarakis, P.M.E. De Bra (Editors.), Proceedings of the Third Workshop on Structural Computing (SC3), Lecture Notes in Computer Science, vol. 2266, Springer, Berlin, 2001, pp. 183–193.

OMG. The common object request broker: architecture and specification. Tech. Rep. 2001-09-34, Object Management Group; 2001. URL http://www.omg.org/cgi-bin/doc?formal/01-09-34

Piaget J. Le Structuralisme. Que sais je, onzième Edition. Paris: Presse Universitaires de France; 1996.

Reich S, Tzagarakis M, De Bra PME (Editors). Proceedings of the Third Workshop on Structural Computing (SC3), Lecture Notes in Computer Science, vol. 2266, Springer, Berlin, 2001.

Reich S, Wiil UK, Nürnberg PJ, Davis HC, Grønbæk K, Anderson KM, Millard D, Haake J. Addressing interoperability in open hypermedia: the design of the Open Hypermedia Protocol. New Review of Hypermedia and Multimedia 5, 1999.

Reinert O, Bucka-Lassen D, Pedersen C, Nürnberg PJ. Caos: a collaborative and open spatial structure service component with incremental spatial parsing. Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia (HT '99). New York: ACM Press; 1999. p. 49–50.

Swick RR. Putting it together: RDF: weaving the web of discovery, netWorker 3.2; 1999. p. 21–5.

Tucker RC (Eds.). The Marx-Engels Reader, Norton, New York, 1978.

Tzagarakis M, Avramidis D, Kyriakopoulou M, Schraefel MC, Vaitis M, Christodoulakis D. Structuring primitives in the Callimachus component-based open hypermedia systems. J Network Comput Appl 26 (2003) this issue doi:10.1016/S1084-8045(02)00064-4.

Tzagarakis M, Vaitis M, Papadopoulos A, Christodoulakis D. The Callimachus approach to distributed hypermedia. Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia (HT '99). New York: ACM Press; 1999. p. 47–8.

van Harmelen F, Horrocks I. Reference description of the DAML + OIL ontology markup language. Tech. rep., DARPA; January 2001. URL http://www.daml.org/2000/12/reference.html

van Ossenbruggen J, Geurts J, Cornelissen F, Hardman L, Rutledge L. Towards second and third generation Web-based multimedia. Proceedings of the Tenth International World Wide Web Conference. Hong Kong, China: International World Wide Web Conference Committee; 2001. p. 479–88.

Wang W, Haake J. Supporting workflow using the open hypermedia approach. In: Nürnberg PJ, editor. Proceedings of the Workshop on Structural Computing (SC1), no. AUE-CS-99-04 in AUE-CS technical reports, Aalborg University Esbjerg. Esbjerg, Denmark: Aalborg University Esbjerg; 1999. p. 12–7. URL http://cs.aue.auc.dk/publications/

Wiil UK, Hicks DL, Nürnberg PJ. Multiple open services: a new approach to service provision in open hypermedia systems. Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia (HT '01). New York: ACM Press; 2001. p. 83–92.

Wiil UK, Hicks DL. Providing structural computing services on the World Wide Web, in: M. Tzagarakis, S. Reich (Eds.), Proceedings of the Third Workshop on Structural Computing (SC3), Lecture Notes in Computer Science, Springer, Berlin, 2001, pp. 160–171.

Wiil UK. Multiple open services in a structural computing environment. In: Nürnberg PJ, editor. Proceedings of the Workshop on Structural Computing (SC1), no. AUE-CS-99-04 in AUE-CS technical reports, Aalborg University Esbjerg. Esbjerg, Denmark: Aalborg University Esbjerg; 1999. p. 34–9. URL http://cs.aue.auc.dk/publications/

Wiil UK, Nürnberg PJ. Evolving hypermedia middleware services: lessons and observations. Proceedings of the ACM Symposium on Applied Computing (SAC '99). New York: ACM Press; 1999. p. 427–36.

Dr Peter J. Nürnberg is an Associate Professor of Computer Science at Aalborg University Esbjerg (Denmark), where he has been working since 1999. Previously, he held positions at Texas A&M University (USA) and Aarhus University (Denmark). His main research interests include structural computing, hypermedia, knowledge management, and digital libraries. More information can be found at http: //cs.aue.auc.dk/~pnuern/

Monica M.C. Schraefel is an Assistant Professor in Human Computer Interaction and Hypermedia in the Department of Computer Science at the University of Toronto, Canada. Before Coming to University of Toronto, she was at AT&T Research Labs in the Online Platform Research Group, New Jersey. Before that, Monica was a graduate student and then faculty member in Computer Science at the University of Victoria, Victoria, BC, where, from time to time, you could catch her with Her Very Hungry Band playing her own brand of folk funk. Papers, further project descriptions and bits of tunes can be found at http: //www.dgp.toronto.edu/~mc