HIVEWARE:
A Synchronous Distributed Context Distributed Content Groupware Architecture

Robert Tischer, BS, MA(equivalent), MS
Hiveware, Inc
Falls Church, VA 22042 USA
rtischer@hiveware.com

Abstract

Cooperative work occurs every day among people using natural language. Implementing a model of natural language would give computer-supported cooperative work (CSCW) research a beneficial new methodology.

This paper presents research that describes the anatomy of natural language context and how the understanding process works; as well as linguistic research that ties deep structure with surface structure, using linguistic signs and not strings of characters. Also presented is a model of language, HIVEWARE, which stands for Hyperstructured Interactive Virtual Environment, and is the operationalization of the CSCW architecture presented herein. HIVEWARE simulates all the saliant features of language and can be used to create and run groupware applications of any kind.

As partial tests of this methodology, two proof-of-concept prototypes were accomplished. A discussion of typical CSCW problems compared to the HIVEWARE groupware application utilizing Microsoft Word shows that the HIVEWARE approach improves upon or even eliminates the CSCW problems in question.

Keywords

computer-supported cooperative work, natural language context, semantic grammar, serverless groupware, SGML, HIVEWARE

ACM Classification Keywords

H.5.3 [INFORMATION INTERFACES AND PRESENTATION]: Group and Organization Interfaces---*Computer-supported cooperative work*

1. Introduction

Natural language (NL) is arguably the best tool people have for working together cooperatively. Instead of using the more typical engineering approach, conducting studies to categorize and evaluate human abilities in terms of machine performance and thereby improve upon groupware applications, this paper postulates that psychological and sociological aspects of NL can be directly emulated on the computer. The goal of modeling NL is to eliminate seemingly difficult aspects of the CSCW architectural task before they can present themselves as a CSCW problem, as well as to accomplish the CSCW engineering task. The result is HIVEWARE, which organizes and focuses group activity supported by the use of computers synchronously harnessed to a semantic grammar fabric, and distributed among participating authors. The result of this methodology is CSCW tools that produce distributed applications that coordinate the atomic actions (not keystrokes) of evolving sets of author's and code designer's creative activities toward accomplishing single domain goals.

There is a reason this approach is not usually taken. Computer science research activities typically do not embrace the study of meaning of language because of the difficulty in defining it. In fact, Noam Chomsky in 1957 declined to use language meaning (semantic) in the formulation of his context free grammar (CFG) theory because he was "not acquainted with any detailed attempt to develop the theory of grammatical structure in partially semantic terms..." [3].

But what if substantive progress in CSCW research were inhibited by the lack of meaning's definition?

Below is a brief description of context theory (Section 2), followed by a description of the operationalization of that theory (Section 3) and examples of two applications (Sections 4). Those descriptions are followed by a comparison of the projected features of the architecture the system described herein would have with respect to the current CSCW research. It will be discussed (Section 5) how a biproduct of HIVEWARE's architecture is that it indirectly obviates whole categories of problems or at least mitigates their effects substantially (e.g., concurrency, responsiveness, consistency, notification, access privileges, HCI, awareness).

2. A Context to Sign to Context Theory of Language

The cardinal aspects of NL context and the understanding process were adequately modeled by addressing the human NL capacities from four distinct viewpoints: 1) how words, things and thing representations are historically created by individuals in a language group (**C**reate); 2) how language seems to

have no lower or upper semantic bound and therefore is always able to describe itself using itself (**R**eflexive); how competent language users can read, write, understand and talk (**E**xpress); and 4) how offspring acquire language competency (**A**cquisition). Hence, the **CREA** theory of context [13]. Part 2 of the NL research ([14] "How the Understanding Process Progresses") models a practical theory of meaning. The combined essence of this research is the definition of meaning being dependent on two criteria: 1) no NL content modeling is possible without a priori context, and 2) no NL context modeling without behavior.

The research points out that a phylogenetic doubling occurred, once for tangible tools, and again for intangible language as tools, hence the twice-doubleness of cognition [9], also known as the ratchet effect [17]. The twice-doubling of tools explains the human exponential cognition explosion.

E is related to **C** in linguistics through the use of the $\frac{\text{signify}}{\text{signified}}$ construct introduced by Saussure at the birth of Linguistics as a field [11]. Semiotics is the academic field for the study of signs, where signs refer to that which does the signifying. Noam Chomsky's Transformational Grammar theory (1960) which says that a sentence's deep structure is a direct representation of the basic semantic relations underlying a sentence which is mapped onto the surface structure is more current. This paper's working NL hypothesis holds Chomsky's deep-to-surface relationship to be, in general, true, but without limiting the surface structure to the generation of sentences composed of strings of characters. A redefinition of the surface structure into Saussure signs is vital.

The deep semantic grammar structure is tightly related to the sign surface layer. This relationship is referred to herein as being homomorphic. Together, these constructs will be referred to herein by the acronym **GSHI**, grammar-sign homomorphic invariant. Essentially, sets of signs and reflexive grammars are postulated as having an onto relationship (refer to ([16], Section **5.2.5 Grammar's Homomorphic Relationship to Signs** for further discussion of a computer science interpretation of this relationship). The well accepted linguistic Sapir-Whorf hypothesis states that one's language determines to some degree our thought patterns. HIVEWARE's grammar-to-sign homomorphic relationship is an idealization of this hypothesis' learning and persuasion influence over the user. Depending on how well the node designers preserve this homomorphic grammar-to-sign invariant, their implementation will serve to teach and persuade the users in how to appropriately act with respect to the group.

The **CREA** [13] and understanding process language research [14] was completed at the University of Copenhagen in 1985. In order to test the resulting CSCW meta-architecture, two prototypes were later accomplished at George Mason University [16] that demonstrated the key aspects of a distributed semantic grammar controlled by a homomorphic sign relationship to the deep semantic structure.

3. HIVEWARE: An Implementation of Context to Sign to Context Theory of Language

HIVEWARE is a meta-CSCW-tool which was modeled on the CREA and Understanding research cited above. HIVEWARE's prominent operational features are expressed by the terms *synchronous distributed context distributed content*.

The present implementation of HIVEWARE is a monolithic software program consisting of a distributed long-running computer process that begins on a single computer. This process lets a semantic grammar designer use ordinary markup language (standard generalized markup language, SGML) to interactively create the root node of a semantic grammar. In markup language terms, the left-hand-side is an element (a.k.a., non-terminal in computer science terms) and the right-hand-side is the content model (a.k.a, production rule). In computer science compiler terms, a ***non-terminal : production rule*** has a ***semantic action***, which contains its code behavior. This triad in HIVEWARE is a thread called a ***node***.

Each HIVEWARE node is a semantic grammar fragment that is a dynamically linked executable, which is compiled from within the context tree using a conventional compiler at each node. Tischer developed the notion of non-sequential parsing, which tied reduction of production rules to a continuous parsing process mapped to Saussures speech circuit [16].

The HIVEWARE implementation operationalizes NL's twice-doubleness creation activity (**C**) by retaining the created elements and behavior in a node tree, such that all subsequent concept nodes can be created iteratively, thus folding each new concept node into all participant's structures so that new authors can create or instantiate new nodes contemporaneously with an always current context.

The result is a hierarchically grown node tree which is replicated at each participant author or subscriber site. Each node is thus continually updated as context and content changes occur. Two or more such nodes are, in effect, a continuously alterable parse tree with evolving node-specific semantic actions that in concert emulate the language context of the authoring group. Like NL, HIVEWARE groupware applications have no central server or management system, and by extension should be scalable to the extent that NL is scalable.

Retained Parse Tree: HIVEWARE does not discard the parse tree as is the case for conventional sequential compiling [16]. Since creating a node in a parse tree is

the major step in a compiler that allows unambiguous translation into target code, and since NL does not throw away its language entities but lets them circulate in perpetuity in the collective minds of each language community [13], HIVEWARE retains its incrementally built parse tree as well. The result is a domain-specific, implemented language, common to a group of people accomplishing a common task.

Grammar Connectors and Occurrence Indicators: Standard generalized markup language (SGML) was used to model the context creation possibilities at any given node in the context tree. A HIVEWARE groupware application is a replicated, synchronous, implemented yet evolving, SGML document type declaration (DTD). SGML is ubiquitous as an international standard and SGML has an advantage over XML in that its DTD allows the expression of non-sequential grammars (content models), e.g., **Fruit : Apple & Pear & Grape** and not **Fruit : Apple , Pear , Grape**. The former, non-sequential representation, adheres to the NL context theory used in this paper.

Connectors (**, | &**) describe how a content model's elements may be instantiated and occurrence indicators (**? * +**) express basic instance cardinality.

A DTD grammar model is static if elements and content models are not permitted to change (e.g. the http protocol). There can usually be an infinite number of grammar variants per DTD. Without being able to change a DTD or ontology's element names, the resulting context would be static. Since **CREA** describes how NL creates and uses context, the static DTD or ontology, and the groupware applications that are dependent on such, would fail the meaning emulation test as not being sufficient. HIVEWARE implements a Variable Semantic Grammar with Variants (VSGV) among participants. The variants are the results of the application of occurrence indicators and connectors. A VSGV is an evolving DTD. In contrast as an example, all HTML pages in the world are based on the same, very simple office memo DTD, that produce tags like <html><head><title> and <body>, etc.

Synchronous Distributed Grammar Context: DTD elements are semantic chunks that are mutually exclusively and hierarchically meted out to authors by authors after creation.

For NL context, a created concept is imperfectly disseminated throughout a language group after its conception. HIVEWARE idealizes this process by allowing authors with context privileges to further subdivide their concept node into subnodes and to deploy them to authors of their choice. The disseminating author retains grant and revoke privileges on descendent nodes and is HIVEWARE's way of emulating the genesis and semantic ownership of new NL entities. The result is a domain specific language, implemented across an evolving set of authors and computers, thus emulating the diachronic aspect of **C** in **CREA**.

HIVEWARE generates domain-specific replicate node trees. It permits the growing of semantic grammars [7] without a server which means that the most appropriate actor to which a grammar node has been deployed, also has the concurrent capability of altering the context in his semantic area. Using Ellis' terminology, HIVEWARE (at the context level) is a communication-oriented coordination system and (at the content level) is a technological group operation [6].

Synchronous Distributed Content: The **E** in **CREA** is emulated by distributing content changes. HIVEWARE idealizes NL's imperfect propagation by making all other participants (authors and subscribers) be observers of **E**'s content changes.

Presentation layer and grammar context: HIVEWARE is an interactive distributed, incremental grammar acquisition system (IDIGA) [7] with distributed semantic grammar and semantic action development. The deep-to-surface relationship between **E** and **C** is tightly operationalized by maintaining a homomorphic relationship between the two and by using standard graphical user interface (GUI) techniques (not strings of characters) to represent the deep structure which unambiguously guide the user. Without this kind of binding, the author user would be unable to make seamless and unambiguous grammar variant and content choices from the surface representation.

Inherited Implementation: HIVEWARE uses the "What You See Is What I See" (WYSIWIS) paradigm. The technological protocol for HIVEWARE groupware is the floor-control-mechanism at the context level which means one person owns a concept (i.e., "has the floor") at a time. This exemplifies NL's **C** behavior in that new concept tools are created by an NL community, but that they are created from one actor at a time for a given point in the context tree.

Social groupware protocols (i.e., mutually agreed upon ways of interacting) are application and even node specific, and are made possible by the inherited implementation nature of HIVEWARE. A node's subnode inherits the implementation of the parent and may be extended by developers (low-level) or tailored by users choosing alternate grammar variants via its surface representation (high-level). The point here is that because of HIVEWARE's inherent inherited implementation capability, any social protocol can locally be altered and projected down the node tree. Compare this VSGV design and implementation with the Wiki phenomenon which relegates social control of the target work to ordinary group dynamics.

Populating new participants: Any HIVEWARE node can be used to construct a new subscriber or author tree. After admission into the group, s/he receives the root node whose initialization code is run. It then receives that node's child nodes, and on down to the leaf nodes in prenode traversal order. This process realizes the **A** aspect of **CREA** and is analogous to a child learning a language in a relatively short period of time.

Standardization: For the changing of context to be a decentralized activity like NL, it is necessary for the meta-grammar (markup language) employed to be internationally standardized, which is the case for SGML. SGML is a closed set of analytical constructs and is in no way meant to be treated formally as corresponding to mathematical expressions. Instead, SGML's use in HIVEWARE attempts to forge (i.e., reify) a subtle connection to the real world of understanding (i.e., **C** to **E**) [15].

The computer language C++ is used to develop node semantic actions for the same standards reason. Thus deploying a node can consist of inherited, runtime linkable shared libraries from the parent in order to realize the fixed portion of the inherited implementation deployment, along with C++ code to be further developed at the new node site. Given the inherited implementation design of HIVEWARE, cross-platform issues can be handled at any context level and automatically disseminated down the node tree.

4. Proof of Concept Prototypes

Fixed Grammar to Surface Prototype #1: Figure 1 shows a fixed semantic grammar[1] prototype with homomorphic sign to grammar enforcement.

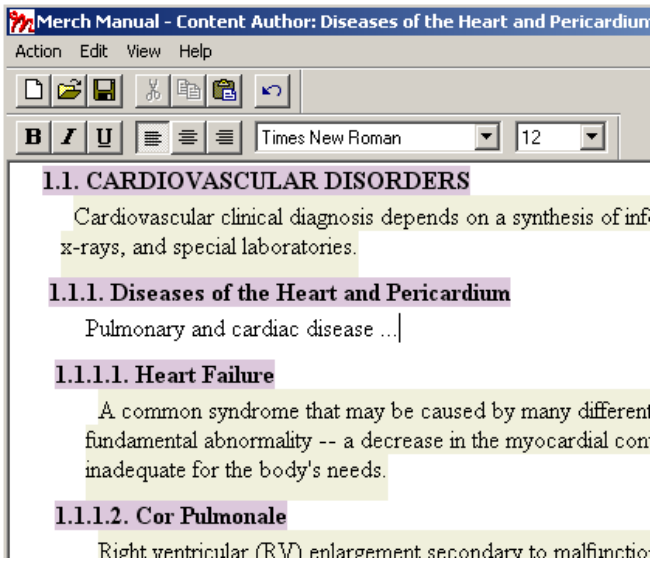


Figure 1 HIVEWARE proof-of-concept word processor example

[1] See YouTube and search using HIVEWARE to view prototype video.

The shading of the section numbers and text and cursor avoidance (cannot be shown) are sign representations of the underlying semantic grammar nodes (not their content) and their associated behavior. As an example, consider this content model:

```
<!ELEMENT Cardiovascular  - -
(HeartPericardium?,ExerciseHeart?,
                AortaBranches?,(#PCDATA)) >
```

Figure 2 Cardiovascular semantic grammar rule

The `Cardiovascular` node's unambiguous sign representation is:

1.1. CARDIOVASCULAR DISORDERS

Figure 3 Cardiovascular's sign representation

Figure 3's content also has a visual cue (shaded text) which unobtrusively notifies the user that s/he does not own that node. The relationship between the shading and its uneditable section text to the `Cardiovascular` element is homomorphic. That is, the user cannot make a choice that is outside the realm of the specified semantic grammar (**Error! Reference source not found.** overt/covert structure cues), which in concert serve to reify the `Cardiovascular` element. This HIVEWARE site is, however, the owner of the `HeartPericardium` subconcept node and the sign representation. White background plus cursor maneuverability indicate the text is read/write. Hence, the grammar-to-sign homomorphic invariant is preserved.

Expression Grammar Prototype #2: The **CREA** model was successfully applied to solve the classical computer science problem of the ambiguous arithmetic expression grammar [16] (e.g., is 3 + 4 * 5 = 23? or 35?) (see Figure 4). Expression grammar fragments were sent at will to various consumers. Note that expression grammar has the difficulty level of being both recursive and ambiguous in the computer science sense. Also note that ambiguity is controlled by grammar-to-sign homomorphic invariant in the form of parenthesis signs and a decorated and maneuver-restrained caret. These computer-controlled signs unambiguously guide the user's choices with respect to the underlying grammar.

E.g., in Figure 4, the carets,  and  , indicate where digits, **+** and ***** can be entered given their position.

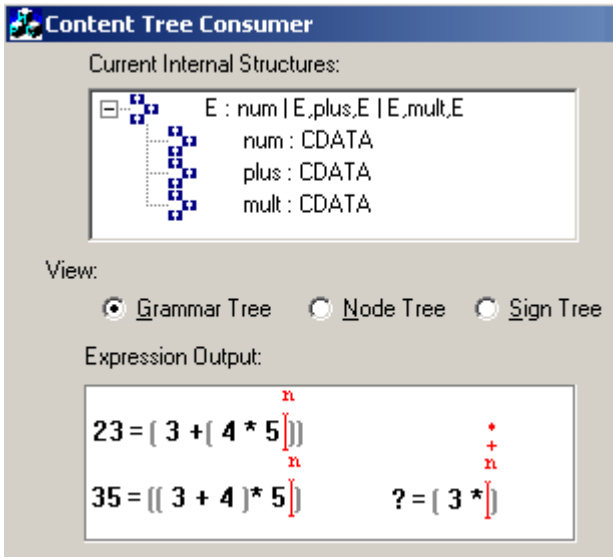


Figure 4 HIVEWARE proof-of-concept expression grammar example

5. Current CSCW Engineering Problems

Concurrency, Responsiveness, Merging: It has been reported that many groupware concurrency approaches such as the floor control mechanism can actually hinder tightly coupled teamwork [6]. The floor control mechanism is similar to database concurrency control so engineering solutions have centered on how granular the locking item should be and the optimistic vs. pessimistic concurrency approaches. With HIVEWARE, the issue is nonexistent since all semantic grammar nodes are mutually exclusively owned and therefore contain no concept of a user session.

Concurrency issues imply a centralized controller designed to maintain concurrency. It has been reported that centralized controllers can decrease groupware application responsiveness [6]. HIVEWARE has no centralized controller.

But eliminating the controller, as HIVEWARE automatically does, typically causes merge issues. HIVEWARE employs pre-content node ownership negotiation which is analogous to NL concept and word creation, and which obviates the competing author scenario. HIVEWARE guides authors, analogous to an organization chart, to their respective functional positions in the node tree, thus leaving negotiation and actual role decision-making to the authors. HIVEWARE does not emulate the human thinking process by offering computer reasoning algorithms as does, e.g., the Semantic Web and Service-Oriented Architecture (SOA). The difference here is subtle but important: HIVEWARE retains intellegent actions humans have performed. It does not try to emulate human reasoning through, e.g., inferencing.

Human-Computer Interaction (HCI) and Consistency: the WYSIWIS paradigm has been reported as being limited because of its inflexibility [6]. However, the study's groupware tool did not support users who wanted to move tree branches. Since the context tree is present at each of a HIVEWARE application's replicates, it will be moveable as a node changes thus eliminating the problem experienced in the study.

Access privileges: groupware's requirements can lead to complex access models [6]. Words (as tools) have a sociogenesis (i.e., the means by which cultural artifacts and practices - including language usage - are created) [17][13]). Not being client/server, HIVEWARE obviates the need for access privileges by only allowing mutually exclusive grammar node creation (the **C** in **CREA**) and deployment in different parts of the tree.

Notification: realtime notification of events, i.e., context and content and node code behavior changes, are modeled in HIVEWARE by pushing the change to all users as it occurs. Obviously, this is the ideal case in a language community and in practice, HIVEWARE applications would need to consider generic ways and means for routing or inserting workflow hops to not only simulate NL dissemination patterns, but to be able to respond to information routing requests.

Data Replication and Consistency: It is common to assume data replication leads to concurrency and consistency issues [19], but when the context is a priori distributed using the floor control mechanism coupled with enforcing a homomorphic relationship between presentation and the underlying grammar, neither problem by design can appear in a HIVEWARE groupware application.

Awareness: Studies report that WYSIWIS lead to increased group awareness and immersion, which in turn lead to improved work product cohesiveness and consistency [10][8].

Insufficient awareness of other's activity was noted in [5] as a problem. A difference in the HIVEWARE's cooperative approach and purely collaborative groupware approaches is, a potential participant in a HIVEWARE groupware application has to navigate through a human decision with respect to the role s/he will play. Being admitted as a member of the group is equivalent to operating a particular post in that context tree. Purely collaborative tools (e.g., Groove) juxtapose people virtually (e.g., videoconferencing, email, IM). This juxtaposition is done without any role decision made at admission time, which reduces group awareness. When a HIVEWARE node appears, it is known that there is a person controlling its content. Consequently, a cooperative CSCW system would have fewer awareness problems than a collaborative CSCW system.

6. Conclusions

This paper describes the design of a system that was motivated by a theory of NL context, the backbone of meaning. Meaning is closely related to context since linguistically, the construction of context over time among people and within the individual, is what gives us our competencies for deriving and generating new meaning [13]. This constitutes a framework for contextual meaning.

Instead of using the engineering architectural model of incremental CSCW improvement, the above framework for meaning was used, thus reifying meaning, to design a candidate CSCW groupware meta-tool, HIVEWARE. This was done by, 1) by emulating NL context and content creation and 2) binding overt and covert graphical surface cues to the emulation of any single-user application today. With this approach, architectural correctness and verifiability are removed from any particular academic discipline's preferred proof technique (e.g., type checking formalism [1]), and placed instead on 1) the adequacy of the NL modeling, and 2) how well that modeling was operationalized.

This paper demonstrates how the typical CSCW issues of concurrency, responsiveness, notification, access priviledges, HCI and consistency were obviated or mitigated by HIVEWARE's design.

Mutually exclusive ownership of syntaxed (i.e., instrumented with connectors and occurrence indicators) semantic grammar nodes, and the use of GUI techniques to realize the homomorphic relationship to their deep semantic grammar emerged as the two main operationalization constructs of HIVEWARE.

In general, the studies reviewed for this paper did not make clear their reasoning for 1) how their groupware designs justified the use or attention to a particular social factor [5][6][8][18][10][19], or 2) why a particular proof methodology was chosen [1][2][3].

7. Further Research

By emulating cooperative NL's two-step process (step 1, create context, step 2: user uses context), HIVEWARE has the potential of separating context coordination from content contribution architecturally. Because of these two steps, future research on HIVEWARE applications would be able to separate technical context building from group dynamics social issues.

Current CSCW research presents no clear distinction between choice of the use of taxonomies vs. ontologies vs. semantic grammars vs. grammar declarations with content models which use connector and occurrence indicator syntax. It is apparently much easier for humans to instantiate than to create. If connectors and indicators represent a vital NL instantiation vs. creation behavior, this distinction should be clarified. A clearer understanding of grammar type differences would facilitate better decisions made by CSCW researchers and implementers when they make their coordination structural choices (e.g. the http protocol). Further research should be conducted on **CREA**'s **C** to describe more accurately how these mundane constructs are related to societal creation of tools and words.

Also, the non-sequential **and** (&) connector was not included in the XML DTD probably because computer science still follows the Chomsky tradition of sequential character parsing. Further research in dynamic semantic parsing with GUI-to-deep structure control is needed to illucidate this possible omission.

8. References

[1] Allen, R., Garlan, D., A formal basis for architectural connection, Carnegie Mellon Univ., (1997), .

[2] Almeida, J. P., van Sinderen, M., Quartel, D. A., Pires, L. F., Designing Interaction Systems for Distributed Applications, IEEE Online 1541-4922, Vol. 6, No. 3, (2005).

[3] Carasik, R. P., Johnson, S. M., Patterson, D. A., Von Glahn, G. A., Towards a Domain Description Grammar: An Application of Linguistic Semantics, Software Engineering Notes vol 15 no 5, (1990), p. 28-43.

[4] Chomsky, Noam, Syntactic Structure, MIT, Moulton & Co. B.V., Publishers, The Hague, 1957.

[5] Convertino, G., et. al., A laboratory method for studying activity awareness, Proceedings of the 3rd Nordic HCI Conf; Vol. 82, Finland, (2004), p. 313-322, .

[6] Ellis, C.A., Gibbs, S.J., Reln, G.L., Groupware: some issues and experiences, Communications of the ACM, Vol. 34, Issue 1, (1991), p. 39-58, .

[7] Gevalda, M., Waibel, A., Growing Semantic Grammars, Carnegie Mellon Univ., Proceedings of the 36th annual meeting on Association for Computational Linguistics, Vol 1, (2005), p. 451-456.

[8] Gutwin, C., Greenberg, S., The effects of workspace awareness support on the usability of real-time distributed groupware, ACM Trans. on HCI, Vol. 6, Issue 3 (1999), pp. 243-281.

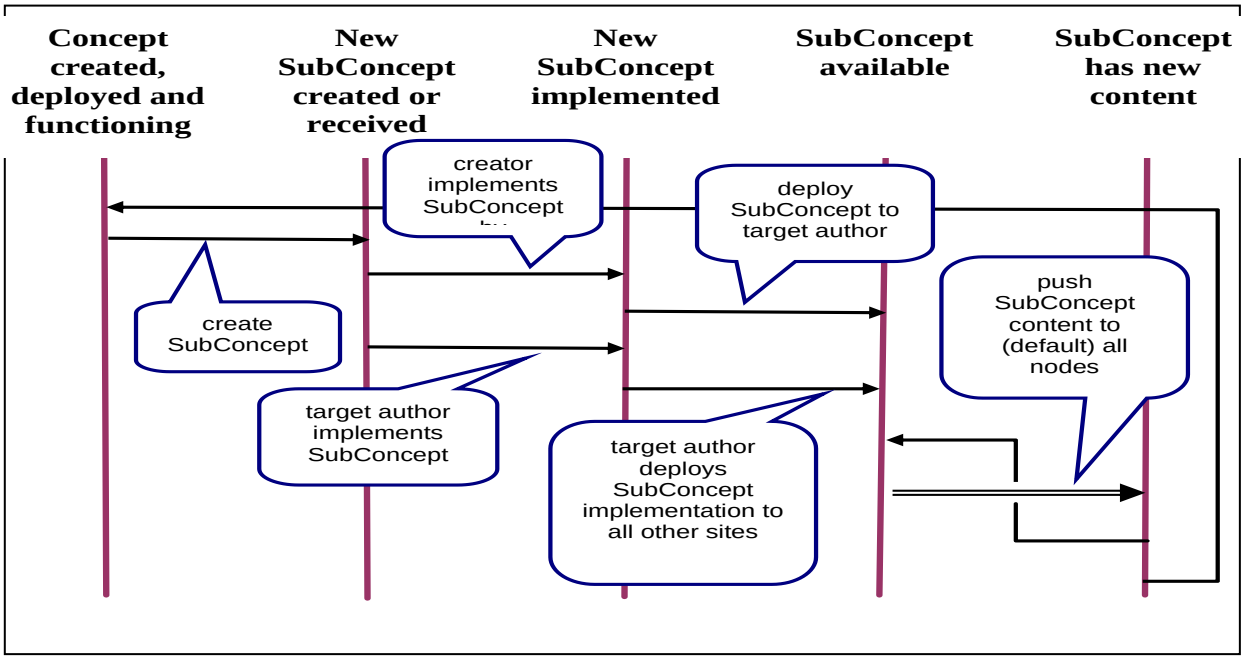
[9] Katzenelson, B., Aspects of the Soul and Spirt's History, Psyke & Logos nr. 2, (in danish) , (1983) p.208-218.

[10] Narayan, M., et. al., Quantifying the Benefits of Immersion for Collaboration in Virtual Environments, Dept. of CS, Virginia Tech, (2005).

[11] Saussure, F. de, Course in General Linguistics, published post-humously in 1916, translated by Roy Harris, Open Court Publishing Company, (1972).

[12] Tischer, R.G. Characteristics of SGML text-entry systems, SGML WG8, (1987), http://y12web2.y12.doe.gov/sgml/WG8/REGISTER/full.htm (available from author).

[13] Tischer, R. G., The Anatomy of Context, MA Thesis in Language Psychology (Part 1) in Danish, University of Copenhagen, (in danish), (1985). See [16] Appendix C for English synopsis.



[14] Tischer, R. G., How the Understanding Process Progresses, MA Thesis in Language Psychology (Part 2) (in danish), Univ. of Copenhagen, (1985).

[15] Tischer. R. G., Theoretical Basis for the SGML Content Model, in The SGML Handbook by Charles Goldfarb, Annex H, Oxford Univ. Press, (1990).

[16] Tischer, R. G., Dynamic Syntax Compiling, MS in CS thesis, George Mason University, (2001).

[17] Tomasello, M., The Cultural Origins of Human Cognition, Harvard University Press, (1999).

[18] Veiga, L., Ferreira, P., Semantic-Chunks a middleware for ubiquitous cooperative work, Distributed Systems Group, Lisboa, Portugal, (2005).

[19] Yang, Y., Li, D., Dynamic architectures: Separating data and control: support for adaptable consistency protocols in collaborative systems, Proc. ACM CSCW, (2004), pp. 11-20, .

9. HIVEWARE apps start blank

A HIVEWARE app starts as an undifferentiated running process and gets more and more defined as the grammar nodes take shape. In contrast, a conventional application begins running all of its functionality at once. You run it and then exit it. A HIVEWARE app is a long-running process that an individual user only exits if he explicitly want to go offline (airplane trip, reboot, etc.). In which case, when you get back on, your HIVEWARE app repopulates from any of the others in the HIVE. All behind the scenes. That is why I was saying that File/New, Exit, Save and Save As don't really have any meaning from a HIVEWARE perspective. In fact, when Word comes up, it will already be populated with some half-finished document that the group is working on. In contrast, Word today comes up blank when it starts.