in: Nicolaas Mars, Towards very large knowledge bases (proc. 2nd international conference on building and sharing very large-scale knowledge bases (KBKS), Enschede (NL), 1995), pp143-155, IOS press, Amsterdam (NL), 1995

Building consensual knowledge bases: context and architecture

Jérôme Euzenat
INRIA Rhône-Alpes
IMAG-LIFIA,
46, avenue Félix Viallet,
38031 Grenoble cedex 1, France
Jerome.Euzenat@imag.fr

ABSTRACT

A protocol and architecture are presented in order to achieve consensual knowledge bases (i.e. bases in which knowledge is expressed in a formal language and which are considered as containing the state of the art in some research area). It assumes that the construction of the base must and can be achieved collectively. The architecture is based on individual workstations which provide support for developing a knowledge base: formal expression of knowledge through objects, tasks and qualitative equations annotated with hypertext nodes and links. It also provides tools for detecting similarities and inconsistencies between pieces of knowledge. These bases can be grouped together in order to constitute a new reference knowledge base. The process for constructing this last base mimics the submission of articles to peer-reviewed journals. This is achieved through a protocol for submitting knowledge to the group base, confronting it with the content of that base, amending it accordingly, reviewing it by the other knowledge bases and finally incorporating it. The system is to be used by researchers in the field of genome sequencing.

KEYWORDS: CSCW — knowledge sharing — knowledge revision — negotiation — protocol — knowledge communication

1. INTRODUCTION

Research activity is principally collaborative: available knowledge is increased by the work of many people in different disciplines and in many places. Collaboration is achieved through one main medium: written books and journals. Thus it takes months and years to be fruitful.

Nowadays, research in molecular genetics aims at understanding and comparing the information contained in genomes. The data involved has two important characteristics: it is growing so fast that it cannot be dealt with manually and it is expressed in such a way that formal computerised treatments can be applied. Thus, an important proportion of people involved in molecular genetics work in "collaboratories": the research teams are located in different laboratories and they use the computer as a medium. It is used for communicating informal text (by mail or ftp), but moreover for communicating the formal results of experiments (through the large general sequence data banks and small specialised ones [17, 24, 10]). We have already developed knowledge management tools which have been used in the building of knowledge bases in molecular genetics [19] and other fields [25]. We are currently involved in the development of a (software) workstation for molecular genetics research in close collaboration with two biology laboratories.

The "computer as medium" idea could be strengthened by extending it towards the knowledge itself. Hence, instead of merely reproducing the paper journals in computers, the principles of scientific journals must be applied to the knowledge formally expressed in a computer. The result should be a consensual knowledge base, i.e. which everybody in a group agrees to be a reasonable state of the art. It can be used for confronting new results and for learning new knowledge. A consensual knowledge base can also evolve with research results. For that purpose, a computer environment called Co4 (for collaborative construction of consensual knowledge) is presented here. Co4 is dedicated to the incremental and concurrent building of a knowledge base organised around formalised knowledge and a set of various annotations (text, bibliography, image, experimental data, etc.) from which this knowledge originates. It provides researchers with support for, on one hand, expressing, annotating and manipulating their knowledge, and on the other hand, submitting it to other people and achieving consensus.

The organisation of the paper is as follows: first the features of a researcher workstation are presented. This is basically an extrapolation from what already exists in biology laboratories. Then, the organisation of a college whose aim — from the computer point of view — is the constitution of a consensual knowledge base is presented. It includes the protocols for consulting and submitting knowledge to the base. The fourth section takes a closer look at the software ground for such a system and fundamental problems which remain open. Finally, an extended discussion places Co4 in the many current trends of knowledge sharing and collaborative work.

2.The researcher's workstation

The aim of Co4 is the construction of a formalised knowledge base: this means that knowledge is expressed in a formal language carrying a precise semantics. The formal structure of the corpus enables the use of consistency checking or comparison tools for helping the process of integrating knowledge into the base and the process of revising the base when necessary. As anyone would agree, this is far too formal and restrictive, so the formal knowledge is connected to informal knowledge (mainly in terms of text and image) structured in a hypertext network. This informal knowledge has two purposes: recording the reasons for acceptance and changes in the knowledge base and adding annotations to the formal knowledge. This section describes the components of a knowledge base and their manipulation by a user. The next section describes how the knowledge bases can communicate with each other.

2.1. The knowledge model

Co4 comes from two experiences with knowledge bases in the domain of molecular genetics. The first one, ColiGene [18], describes the regulation mechanism of gene expression in the *E. coli* bacterial genome. The second one, MultiMap, allows to describe and to manipulate mammalian genomic maps at the cytogenetic, genetic and physical levels. The researcher's workstation is an assistant to the researcher. It helps the description of knowledge and also its discovery by automating the more repetitive tasks of genome research. The design of these two knowledge bases has led to the identification of four types of knowledge to be represented:

- *Descriptive* knowledge on the biological entities involved is represented in an object-based knowledge representation system. This enables the representation of classes of objects (e.g. genes), subclasses (e.g. protein genes) and the identification of an object as belonging to such a class.
- *Methodological* knowledge specifies the ways to select and link up methods for a given task. It is represented through a task management system able to integrate, represent, process and monitor the many computer programs for analysing the results of experiments.
- *Behavioural* knowledge, which has not been introduced in these two knowledge bases, concerns the modelling of dynamic phenomena, such as the dynamics of gene inhibition or activation, through a qualitative modelling system. Such kind of knowledge has already been used for representing metabolism [10].
- *Textual annotations* on the various objects and tasks involved are achieved through a hypertext system which connects hypertext nodes with the components of the descriptive and methodological knowledge. It allows browsing among texts, objects and tasks (see figure 1).

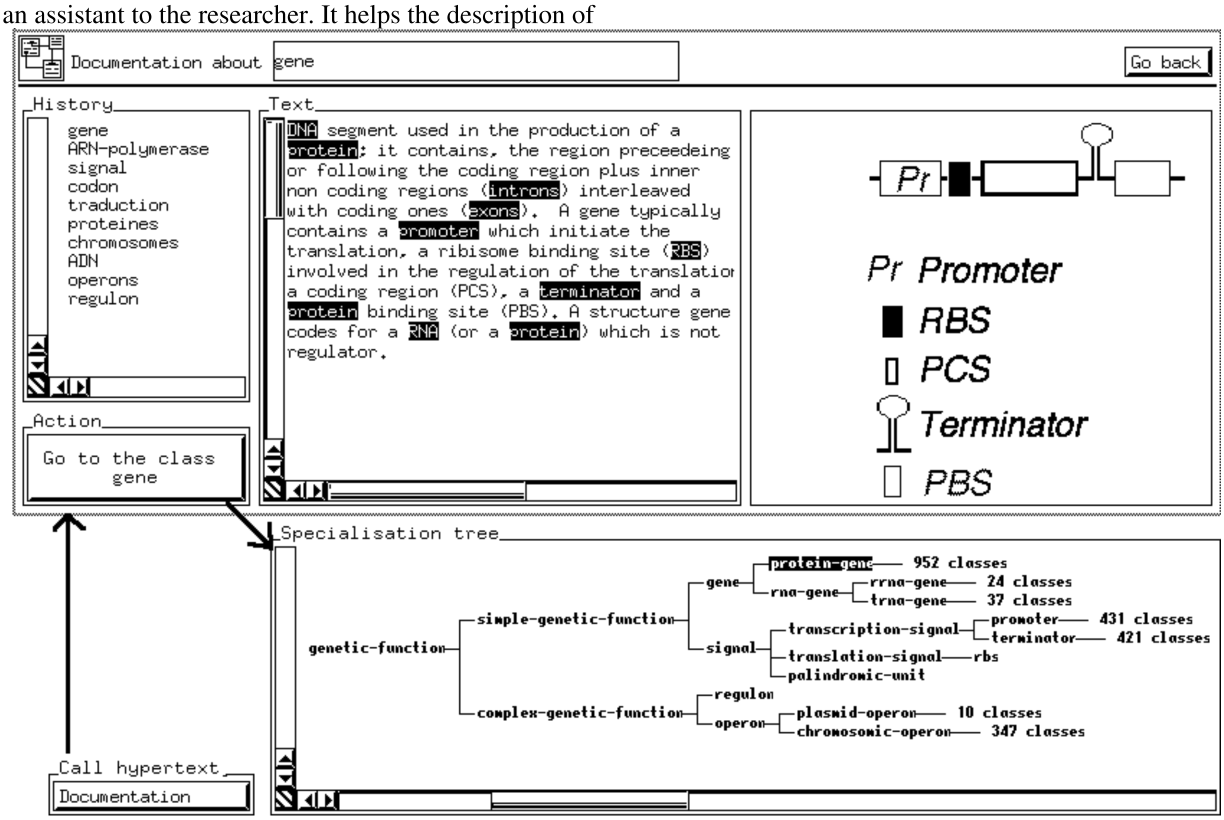

Figure 1 (adapted from [19]): The navigational interface using hypertext. The user has selected a class in the class taxonomy (upper window) and has consulted the associated node of the hypertext (lower window). Some of the words of the text are links to other nodes. From a hypertext node, it is possible to come back to the entity it is attached to. The example is drawn from the ColiGene knowledge base, in which only the highest levels of the specialisation graph have been annotated, essentially for pedagogical purposes.

The knowledge, both formal and informal, is stored in an object-based knowledge representation system called TROPES [14]. It comes with a clear and simple formal semantics and tools for classification, constraint management and task processing. In particular, it is able to organise objects into multiple separated taxonomies and allows the user to work on a subset of these taxonomies. It thus answers to the need to express several viewpoints on object classifications [17, 10]. TROPES precursor, SHIRKA, has been connected to a hypertext management system [8] and it is planned that it will soon be the same for TROPES.

2.2. Software architecture

The architecture of the Co4 workstation is as described in figure 2. It is made up of three main layers:

- A user interface allowing the researchers to communicate with their knowledge bases and with other knowledge bases;
- The knowledge base itself which provides support for storing formal and informal knowledge, detecting inconsistencies in formal knowledge, returning possible modifications in the formal knowledge and managing the dialogue with other knowledge bases.
- The communication layer which provides software and hardware facilities for communicating with other knowledge bases.

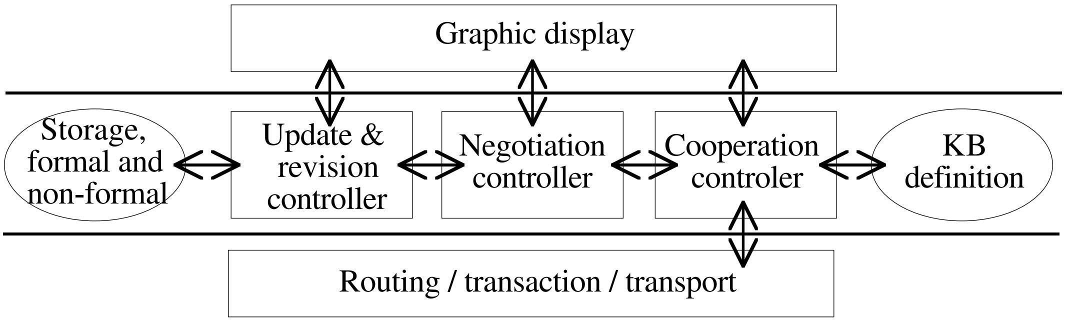

Figure 2: The three-layered software architecture. Each box represents a software module, each circled unit is a data/knowledge repository and each arrow represents the call of a program functionality.

For the sole manipulation of the knowledge base, the user only accesses the revision and negotiation controllers through the graphic interface (see figure 3). The other components are used in the cooperative process; they are included here for completeness. However, the knowledge management system is accessed through an application programming interface which enables queries and tentative modifications of the base from both the graphic interface and messages coming from other bases (see §3).

2.3. The user and the base

The present section shows how the user can interact with the knowledge base. It can seem obvious that the user can query and modify the base and that modifications must leave the base in a consistent state. However, the way this is achieved in Co4 is particular since it must work when the researchers want to communicate a part of their bases to another base. Thus, the way individual stations work is useful for both the individual and the cooperative modes. The process is presented in figure 3.

The consultation or modification are ordered through the graphic interface and directly processed by the revision controller. However, in the usual mode, the modification is prepared by a confrontation query which asks for a comparison of a new piece of knowledge (made of objects, tasks, hypertext nodes and qualitative equations) and the knowledge base. The comparison of a corpus of knowledge with another results in a report about what is different, what is the same and what is contradictory. If the piece of knowledge does not contradict the knowledge base, it can be submitted for integration. If it contradicts it, the researcher can modify it in order to fit the group base. Concerning the informal documentation, if the user wants to create a hypertext node for instance, it is possible to detect if a node with the same name already exists and to negotiate its modification.

A first requirement for a formal knowledge base is consistency. Consistency is here defined with regard to what the system can deduce to be consistent or not: in typical object-based representation systems, a class whose extension is logically reduced to the empty set is inconsistent and an error is raised by the system. Apart from consistency checks, the system is able to deal with sophisticated queries asking if a piece of knowledge is redundant, subsumed or similar (w.r.t. some distance) to a part of the knowledge base. These queries are subject to limitations drawn by the expressiveness of the knowledge representation language and the expected degree of completeness of the answer.

When a change is attempted, it first goes through the update and revision controller which determines if the change does not introduce inconsistencies. The organisation of Co4 is particular in that the revision controller, which is usually a part of the knowledge base management system, has been detached from the knowledge base itself. The revision controller should be able, given an operation on a knowledge base, to manage it in the most consistent way. If the proposal is consistent with the base, the change is committed and the knowledge base is simply modified. The controller also records all the modifications committed directly by the user in order to be able to transmit them later to the group base (the modifications must be recorded together with their rationale, etc.).

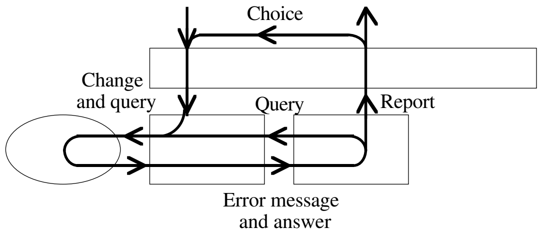

Figure 3: Interactions between the users and their bases. The arrows represent the flow of information: the user can query the knowledge base and get an answer or a report. The report can be made of a set of alternative changes to apply to the base in order for the user to achieve a modification of the base. This requires a dialogue between the revision and negotiation controllers for determining what are these possible changes.

When Co4 detects some problem (misspelling, typo, inconsistency) it raises an error which is transmitted to the negotiation controller. The negotiation controller then opens a dialogue with the update controller in order to establish the putative causes of each error and the possible repairs. Then the negotiation controller is able to submit these diagnoses and repairs to the user through the graphic interface. It is up to the user to decide which one to apply (or to retract the tentative change).

The behaviour and decomposition of the architecture is directed by the wish to provide the most efficient response to a tentative change of the base. It is based on the idea that when users ask for a change, they express the will to see it committed. Thus if this change is not immediately possible, Co4 must propose the best way to make it possible while preserving the majority of the base. This enforces the opportunities for the change to be accepted by the other partners when it is submitted to the consensual knowledge base. The same holds true for propositions of the system concerning similarity.

Moreover, in the context of the submission of knowledge to a collective base, the architecture allows to ask for a report and help from that base. This is presented below.

3. OVERALL ARCHITECTURE AND PROTOCOL

The primary aim of Co4 is the construction of a consensual base. The principles underlying Co4 are derived from those of peer-reviewed journals: before being introduced in a consensual knowledge base, the knowledge must be submitted and accepted by the community. This requires submitting knowledge to the base, letting it be reviewed by the other participants and accepting or amending it according to their reactions. The informal knowledge is also subject to submissions, reviewing and so on. At the end, it is intended that the knowledge stored in a consensual knowledge base be safe enough so that anybody can use it confidently and easily.

In this section we emphasise the collaborative facilities offered by Co4. The organisation of a college of knowledge bases is first presented. Afterwards, the interaction of a workstation with the consensual group bases is detailed before turning to the protocol implemented for dealing with knowledge submission and negotiation.

3.1. The network of knowledge bases

Co4 is made of a set of knowledge bases. Any cooperator is viewed by the system as a knowledge base. Knowledge bases are organised into a tree whose leaves are user knowledge bases and whose intermediate nodes are called group knowledge bases (see figure 4). Each group base represents the consensual knowledge shared by its sons (called subscriber knowledge bases). Each knowledge base can subscribe to only one group. However nothing prevents a human researcher from creating several knowledge bases (maybe subscribing to different group bases) representing different research trends, and nothing prevents anyone from transferring knowledge from one base to another. Also, nothing prevents several physical users from sharing the same base.

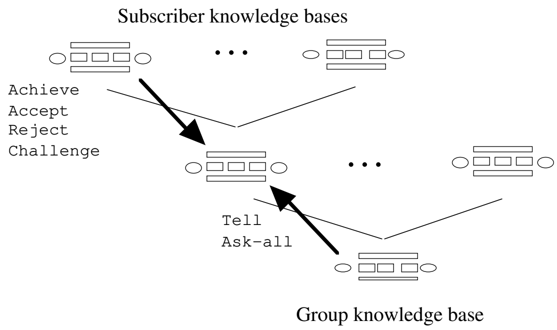

Figure 4: The hierarchical architecture and message flow. The knowledge bases are organised in a tree whose leaves are individual knowledge bases and nodes represent the consensual knowledge of connected individuals. The downward types of messages include the submission of a proposal and the reports of approval, rejection or alternate proposal about a submitted proposal. The upward messages include the broadcast of accepted proposals and the call for comments (ask-all) about a submitted proposal.

The knowledge bases are linked together in such a way that a particular knowledge base knows its subscribers and its group base. To its subscribers it mainly sends messages for broadcasting a change accepted by everyone and calls for comments in order to establish whether a change must be committed or not. To its group base it sends changes that it wants the group base to integrate. Of course, as a group base, it also receives changes to commit and as a knowledge base it also receives calls for comments and change broadcasts.

Each of these knowledge bases are made of the same structure and the same software. The main difference between group bases and researcher workstations is that the former are completely automated and only respond to the stimuli from other bases.

3.2. The user and the group

The user of a workstation can subscribe to a consensual knowledge base. This is achieved very simply through the knowledge base definition controller which manages the description of the group to which the knowledge base subscribes (and for the group bases, the set of subscribers). This base definition enables the communication layer to route queries from one base to its group base and its subscribers. It is intended that it also describes the topics which the knowledge base is interested in, etc.

As soon as the base is part of a group base, it receives the complete content of that base (to which it is supposed to subscribe), it is entitled to give its opinion on all submissions currently under examination and is entitled to submit knowledge. The interesting point is the submission of knowledge, so let's see what happens.

When the researchers are confident enough with the specifics of their knowledge base corresponding to some research results, they can submit them to the group knowledge base to which they subscribe. This is achieved by circumscribing the submitted part (which can include hypertext annotations justifying them) through the graphic interface and calling the submission procedure of the negotiation controller. In order to complete the submission message, the negotiation controller collects the sets of differences between the consensual group base connected and the selected changes (they are logged in by the revision controller) and sends them to the group base. Usually, the group base, through its own revision and negotiation controllers issues a report describing how the submitted knowledge can be added to the group base. Thus, as usual, the user can choose a better (and consistent) way to achieve the submission. This proposal will be submitted to the other subscribers and committed if it reaches consensus.

As a subscriber of the group base, the user also receives the call for comments issued by the group base in response to the submission (by another user) of some material. The users can read the submission or play it in their own knowledge base by submitting it to the revision controller. This can result in a favourable report or an inconsistency detection that can be used by the user for issuing an alternate proposal. In response to the call for comments, the users must answer by one of the following: accepted when they consider that the knowledge must go in the consensual knowledge base, rejected when they do not, and alternate when they propose another change.

When the group base has gathered enough comments, it integrates, or not, the change in the base. The change being now consensual, it is broadcast to all subscribers. It may happen, however, that the research they are currently involved in contradicts what is in the group base. So the users can refuse the new knowledge (just as they can also modify parts of the group base knowledge in their local base) which is then stored in a change logbook for further change submission.

The fact that anyone can maintain a knowledge base as different from the consensus, allows obviously the exploration of alternate research paths. But on a more basic ground, it enables the communication, negotiation and acceptation to be asynchronous. This, in fact, reproduces the way papers are submitted, discussed and accepted or rejected in a scientific journal: the reviewer can take time for carefully examining a proposal since this will not stop the work of the base which issued it.

3.3. The submission protocol

The words consensus, message, etc. have been used freely so far. A protocol has been established for the communication between the knowledge bases. It is implemented in the cooperation controller of the knowledge bases. The protocol is very simple, since it only reproduces what happens for paper submission to journals plus the management of new subscriptions. The messages are expressed as a collection of speech acts: achieve (submit a proposal for inclusion into the consensual base), ask-all (ask for the acceptance of subscriber bases, call for comments), accept (accept the insertion of the piece of knowledge), reject (reject it), challenge (submit a concurrent proposal), deny (the submitter retracts the submission), tell (send an accepted proposal to each subscriber base). It seems quite general since the performatives can be found, for instance, in concurrent software engineering [15, 26].

The consensual aspect is dealt with through the acceptation of proposals which achieve acceptance by all of the subscribers and the rejection of other proposals (for instance, in the context of genome sequencing, a consensus map is a map that people involved in the research field think correct). Consensus could be replaced by some other definitions (like majority or intersection, see §4.1), but it has been retained for two reasons: (1) it enjoys interesting formal properties (if a consensual base contains only knowledge which is accepted by all the subscribers, this remains true if subscribers are added or retracted), and (2) it should lead to the discussion of proposals — not only conflicts — and thus the collaboration of the researchers.

In a first version of Co4, the group base applies the non destructive modifications without discussion. These modifications are always possible in the group base, since they are in the subscriber's base which contains it. In the case of destructive modification, a call for comments, identified by a unique number (surrogate), is issued towards all the subscribers. Among the answers provided by the subscribers, three cases may happen:

- They all agree that the modification must be accepted, then the modification is committed into the group base and broadcast to all the subscriber knowledge bases;
- One of them rejects the proposal, then the changes are not committed and the comments provided by the rejecter are sent to the submitter (the call for comments is discarded in all the subscribers knowledge bases);
- One submitter sends an alternate proposal, then the call for comment is replaced by a call for comment about all the proposals available (those who already

accepted the change, are asked to consider the new proposal and to answer again).

It also can happen that the submitter retracts the proposal thus leading to the retraction of the call for comments from all the knowledge bases.

The protocol is made of a set of rewrite rules which state what a knowledge base must do when it receives a particular message from another knowledge base. Each rule specifies what happens when some performative is received by a knowledge base, for instance:

$$\frac{s \longrightarrow \mathrm{reply}(n,\mathrm{accept}) \longrightarrow g}{\begin{array}{c} C:=C-\{\langle n,\mathrm{achieve}(p),1\rangle\}, \\ K:=K+p, \\ g \longrightarrow \mathrm{tell}(p) \longrightarrow S \end{array}} \ \langle n,\mathrm{achieve}(p),1\rangle \in C$$

means that upon arrival of the last “accept” for a proposition with surrogate *n*, the group base discards the structure recording the call for comments process, adds p to its local knowledge base and broadcasts it to each subscriber. The group bases blindly apply these rules and the final decision comes from the researchers who are the holders of the leaves of the architecture. When a message is issued by a group base whose subscribers are also group bases, these last bases only dispatch the messages to their subscribers. The protocol is presented in figure 5 as a finite state automaton. It is worth noting that the protocol is (1) asynchronous, so that several proposals can be in different states concurrently, (2) parameterised by proposal p, so that the real situation in Co4 is the Cartesian product of automata corresponding to all the proposals, and (3) abstracted from the status of each individual knowledge base with regard to the protocol. Some properties of the protocol can be proved (under additional assumptions): whenever a proposal is submitted, it reaches in a finite time the status of accepted or rejected, the protocol always takes into account the opinion of a reviewer (it never waits for a report when it is established that the proposal is to be rejected, etc. — see §4.3).

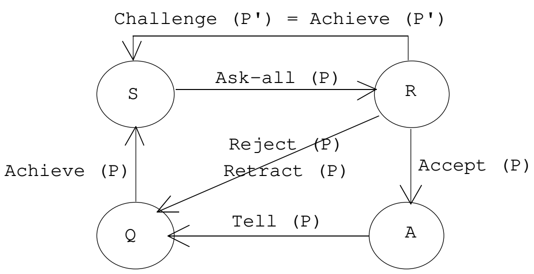

Figure 5: The automaton corresponding to the submission of proposal P at the scale of the whole system of group base plus subscribers. It reproduces four states of publication submission: initial state (Q), submitted (S), under review (R), accepted (A).

The protocol is handled by the cooperation controller which acts differently on an individual base and on a group base: in the first case it submits the changes to the user interface while in the second one it dispatches them to all of its subscribers. The cooperation controller is also in charge of the management of the negotiation for applying a change in a group knowledge base (emitting the call for comments, receiving the answers, managing the alternate proposals and committing or retracting the changes). The negociation requires the identification of the change by a unique number, the count of positive answers, the management of alternate proposals and retractions of a proposal. It also requires the recording of the process in order to recall the reasons why some change is made or not.

4. SOFTWARE TECHNOLOGY

The development of the platform presented above is not yet complete and the representation of genetic knowledge is not a simple task. However, the technology for object based-knowledge representation is well known, and similar experiences that we have had, lead us to be confident of its achievement. Thus, the present section focuses on the specific problems concerning the cooperation between distributed knowledge bases. To our knowledge, the technology is not already established for such a proposal; thus the Co4 proposal is an occasion to explore solutions for two particular fundamental research areas: assisted revision in knowledge bases (§4.2) and cooperation protocols (§4.3). Problems and solutions are presented below, but keep in mind that they should meet acceptance from the users before being declared successful. So, the properties of the architecture enabling a smooth integration of facilities are first presented (§4.1).

4.1. Modularity and genericity

The knowledge base architecture presented above included two particularly interesting design choices: it is the same for all of the bases, so they can be made out of the same software packages, and it is described in a modular fashion, so that it can be used with a very raw system or a very complex one. This leaves the door open for a trade-off between the complexity and power of each algorithm. This is very precious during Co4 development since it allows us to start with very simple components and to enhance them progressively.

For instance, the knowledge repository can be a simple hypertext system which identifies nodes by a reference and raises an error when two nodes have the same reference (the simplest system is a text repository with the UNIX “diff” and “patch” utilities). The *degré zéro* of change control consists in routing error messages to the user, in order to have the error corrected.

Such a system can be enhanced, for example, in the framework of a logic clause representation system (say Prolog) by a revision module able to abduct the reasons for the errors and to display them to the user who can take the corrections into account. Going further in this direction the revision module can be a knowledge base combining algorithm able to automatically propose corrections (e.g. [2]).

Finally, the protocol can be easily replaced by another one, based for instance on a vote of the majority of the subscribers, on the intersection of all of the knowledge base or on economic decision making (for determining the introduction or not of knowledge into the base). The protocol is also designed independently of the content and language of the knowledge base. This enables to use it for different pieces of knowledge (hypertext or classes as well as tasks or equations). One could imagine to refine the protocol for dealing specifically with these expressions. However, the specific aspect is actually assigned to the revision module which issues the reports.

So, the architecture and the protocol provided above are independent of the knowledge repository and the modules which manage it.

4.2. Assisted revision in object based knowledge bases

The aim of the update and revision controller is the modification of the knowledge base. It is usually a part of the knowledge base able to perform an authorised operation on a knowledge base or to raise an error. The revision controller should be able, given an operation on a knowledge base, to resolve it in the most consistent way (this covers truth maintenance or, when an error message is raised, finding the best way to achieve the operation). The aim of Co4 will be to propose meaningful ways to realise it, the last word being that of the user or the subscribers.

Assisted revision of knowledge base — i.e. the ability of detecting error messages and proposing consistent changes — is related to the problematic of knowledge base revision. It is a real enhancement with regard to current knowledge base management systems and is required by the submission of the knowledge of a particular base to another.

Knowledge base revision has been introduced in [1]: from a set A of axioms which deductively leads to a theory Th(A), let the operation $\perp$ be such that $A\perp p$ is the set of maximal subsets of A such that $p \notin Th(A)$. Retraction (–) and revision (+) operations are defined such that $A-p \in A\perp p$ if $A\perp p \neq \emptyset$, and $A-p=A$ otherwise. A set of eight postulates defines what are the possible retraction operators ($A\perp p$ is deductively closed, does not entail p, contains Th(A) if $p \notin Th(A)$, etc.). These postulates can also be defined for revision.

Syntactically, existing knowledge representation systems are already capable of inference, consistency check and even revision since it is possible to add new objets (or classes or field values, etc.). Thus this framework can be applied to object-based systems: the set of axioms (A) is the content of a knowledge base communicated by the subscribers, the theory (Th(A)) is what is deducible from A.

The decision problem on what is deducible or what is inconsistent is computationally language dependent. In TROPES, it seems that it is decidable but that some predicates such as classifiability should be at least NP-complete. In object-based knowledge systems, the foreseen revision operators should have the following properties:

- They strongly take into account the syntactic structure of the object-based representations, which defines the revision rules for the syntactic operators over objects (a class is less often modified than a simple instance for instance);
- They correspond semantically to the postulates for preferred revision — the model preference being defined by a relationship between them — allowing the definition of a relational revision operator [1].

TROPES is able to signal errors. This is the basis for update and revision. These errors are transmitted to the negotiation system (errors are not just messages, but structured objects enabling the identification of the error). This is systematically used in order to detect what are the causes of the error and then what are the possible alternative changes which may be issued. For instance, when an overly large domain is given to a field in a class, object-based systems issue errors such as “your domain includes values which are not possible for the field”. However, the solution can be the restriction of the given domain or the enlargement of the domain for the super-classes of the class (there may be other errors: the class is not the right one, the field is not the right one, etc.). A system like TROPES must be able to find these simple causes for errors and to prepare a repair action.

More operationally, the aim of such operators is:

1) localising quickly all the sources of inconsistency;
2) finding repair solutions to these inconsistencies;
3) ordering these sources and solutions with regard to their syntactic nature;
4) presenting them to the user.

At least it is not intended that Co4 applies the revision by itself, but rather that it helps the researchers to easily revise their knowledge when it is inconsistent. In the framework of such a system, there are strong constraints of speed and intelligibility. It seems that these constraints can be satisfied with the help of the use of interaction which allows fast feedback from the user and the constraining structure of objects (which limit the exploding character of abduction and can reduce the size of the search space).

Another promising idea is the revision operation based on what a the users know and what they are interested in. TROPES already supports, through viewpoints, the selection of a particular perspective on knowledge. Ordering the topics of interest has already been used for expressing the semantics of revision with several people involved (in the expression of knowledge) [3].

4.3. Cooperation protocol

The cooperation protocol is based on the architecture of the knowledge bases (which ones are the group base and which ones are the subscribers) and a complete set of behaviour rules [5]. This protocol has several properties:

— there is no need for human intervention in the group base;
— there is no message but from group base to subscribers and back;
— each decision has been approved by all the subscribers (recursively for a group of group bases).

The actual protocol is routed automatically (once a knowledge base has subscribed to another), the performative and content levels are interpreted automatically in the group bases and the performative level is automatically interpreted by the individual knowledge bases (however, the system asks the user before committing these performatives).

The set of messages sent from one base to another are expressed through a speech act (loosely inspired from “speech act theory”). Speech acts are used here for building a set of relevant “artificial” acts rather than trying to understand the “natural” speech acts occurring in ordinary conversation (the same way the idea of a grammar has been borrowed by computer science from linguistics). Thus each particular act carries a precise semantics (taken as the goal of the sender). The notion of a speech act has several advantages for the particular architecture presented here:

- It allows the separation of knowledge from what its use (add or retract it from a particular base, for instance);
- It is independent from the knowledge representation language and the protocol can thus be expressed abstractly and the library implementing it can be used with other systems;
- It allows a speech act to refer to another speech act (retracting a submission, for instance).

The inter-base communication uses KQML (Knowledge Query and Manipulation Language [6]) which has been chosen because, in its early stages, it clearly distinguished the three levels required by Co4:

Communication: to be used for the communication layer in order to route the message;

Performative: to be used by the negotiation and revision controllers in order to know what kind of action the message is intended to achieve;

Content: to be used by the knowledge base management system.

While building the protocol, some formal properties are required and ensured. For instance, under the following assumptions:

— there is always, in a finite amount of time, an answer to a query (to an individual base),
— there is not an infinite number of alternate-proposal for a proposal, and,
— different proposals are independent,

each submission reaches the accepted or rejected status in a finite amount of time and this status is such that a submission is accepted if and only if it agrees each subscriber. The two first assumptions are part of the fairness assumption of Co4 while the latter requires taking into account the interaction between proposals.

Difficulties with the cooperation protocol come from the fact that it must be closely suited to the needs of cooperative knowledge base building. Otherwise, subscribers would work around it. The ideal protocol should be mechanically interpretable (at least at the performative level) and rich enough for covering adequately all of the needs. The first requirement has been successfully achieved through the construction of an automatic group base protocol [5]. However, our present protocol is very simple (only 33 rules) and must be enhanced through experience and some assumptions may have to be relaxed. For instance, it does not take into account the demand from a reviewer to clarify some point (this is planned but informal communication is not very well taken into account by KQML).

5. DISCUSSION AND RELATED WORKS

So far, the Co4 system has been presented as an extension of existing systems and references to similar systems have been avoided. In this section, Co4 is first placed in the many taxonomies of collaborative systems. Then the system is compared with the many current works on knowledge sharing. Co4 is particular in several concerns for knowledge sharing and collaborative work, thus the third part is dedicated to the explaination of the principles governing Co4 and of why they make it close to some systems.

5.1. What is Co4?

Co4 can find its place among the many taxonomies of CSCW systems. With regard to [20] it has asynchronous indirect interaction, system based coordination, data hiding and is collaborative aware and technically non flexible. While cooperative, Co4 dictates the way the cooperation is achieved. This takes into account the usual context in which research is carried out (as opposed to “is a general purpose support for cooperative work”) in order to deal with usual attitudes as advocated by [11].

Co4 allows the coordination of people for publishing results they consider as achieved. So, as presented in [13], it is only tailored for the execution stage and enables information sharing and activity coordination. As such, it does not aim at supporting the organisation of collaboration on a precise topic by planning experiments and analysis. Extending Co4 is possible, but would require a very different interaction protocol such as the contract net protocol [21].

According to Ellis [4], Co4 would not be a groupware system *stricto sensu*, but could be classified under coordinated multi-user editor. By opposition to Gibb’s definition of groupware, people do not really share a common environment since they work on their own knowledge base. But they share the goal and the content of the consensual knowledge base.

Co4 shares many features with the coordinator [26]. The coordinator is a system which allows people to submit requests and offers to others who can answer by declining, accepting or proposing an alternative. The system is able to store these proposals and to manage the state of each proposal. The main differences lie in the goal of building a consensual knowledge base (common to each individual), the formal treatment that Co4 can apply to knowledge and the hierarchical construction of knowledge bases (and hence of the communication) instead of peer-to-peer communication.

5.2. Co4 and knowledge sharing

The knowledge sharing idea has been promoted as a way to avoid constructing multiple knowledge bases and multiple reasoning systems about the same domain. It defends, instead, the idea of sharing this achievement by either merging the knowledge of several bases into one (knowledge sharing approach [16]) or submitting the problems to several accessible knowledge based systems (software agent approach [7]).

There are some hints that knowledge sharing cannot be achieved without sharing the building of the knowledge base itself. Knowledge sharing requires not only the intelligibility of the knowledge description language semantics, but also the agreement on the meaning of concepts of the domain. A project for sharing the construction of a knowledge base has already been described in [19]: it aims at helping scientists to build a consensual knowledge base about their scientific research domain. In this context, shared with the knowledge medium idea [22], producing and maintaining the knowledge base constitutes an end in itself. Such a framework can also be used for huge scientific projects, distant collaborative works or knowledge elicitation in order, for instance, to achieve a corporate memory.

The Co4 approach differs from the software agent approach because it considers a general architecture and the homogeneity of agents and from the knowledge sharing approach because it assumes that knowledge cannot be shared unless the elaboration process has been shared. The fact that all agents share a common goal represented by their common knowledge base is another unusual characteristics. This led to the design of a very specific interaction protocol and the enforcement of consistency in Co4.

However, the Co4 principle is similar to that of the SHADE project [9] which has one foot in each world. The SHADE project builds a knowledge medium which aims at supporting the collaborative design of an artefact; in Co4 the artefact is the knowledge base. Such a system must enforce both the consistency and the agreement of everyone (human or software agent) involved into the design process. However, there are several technical differences between both systems:

- The SHADE project uses pre-existing knowledge bases (in the knowledge sharing fashion). It thus puts less emphasis on knowledge base revision than Co4.
- In SHADE, agents can be very different while in Co4 all agents are equal (peers) and play different roles (submitter, reviewer, etc.) depending on the situation. The variety of agents constrains to take into account a variety of behaviours (which changes to notify, etc.) and requires tools for expressing how to deal with them (publication of interest) which have not been considered here.

The agents (bases) of Co4 are very structured since they already share their goal (establishing of knowledge base) and a mode of organisation (subscription tree). The society as a whole has to process a contract (building a consensual base by submission) by opposition to establishing a plan for achieving the goal. It does not have resource allocation and coordination problems: the only problem is communication. This multi-agent architecture is like the federated systems of [7], in which agents communicate indirectly through facilitators. Here, each agent is connected to the other through its communication software which is a very specialised facilitator: only groups accept subscription and the communication is enabled only from one group to its subscribers and from the subscribers to the group. The group bases constitute the mediators between the actual users. Their routing level is like both a KQML router and a very simple KQML facilitator; as a consequence, each base has the same router and facilitator components (at the moment they have two: one for sending messages to the group base and one to send messages to its component bases). This is a divergence from the KQML facilitators which are considered as knowledge bases themselves able to dialogue with any other knowledge base.

In the scientific community metaphor [12], close to the software agent approach, the emphasis was on solving problems rather than sharing knowledge. So the aspects of the building process of a corpus of knowledge have not been considered. However, the basic properties of the system (commutativity, monotonicity, pluralism and parallelism) hold true for Co4.

5.3. The Co4 principles

Co4 is here presented in terms of a set of principles which provide the background of the system. These principles allow us to stress the differences between Co4 and other efforts concerning knowledge sharing or collaborative processes.

5.3.1. Uniformity

Co4 is made up of a set of knowledge bases. Any cooperator is viewed by the system as a knowledge base. For simplicity, all the bases are equipped with the same software. Something interesting with regard to other cooperation schemes (and mainly the one used in the software agent trend) is that the knowledge bases have the same description language (this is to be contrasted with other approaches, in which agents are seen as knowledge bases but their languages can be heterogeneous). This simplifies the implementation and use of the system by allowing to stress the collaborative aspects, but, above all, this ensures that people are talking about the same representation formula instead of a translated item. Of course, each base cannot have its own representation system. However, the use of a language independent from the knowledge representation language (KQML), preserves the possibility to use heterogeneous representation languages (but communication between bases expressed in different languages is, to our knowledge, far from achieved). One can easily imagine a particular agent able to translate the content of some sequence data bank into TROPES and able to submit it to Co4.

5.3.2. Consistency

The knowledge stored in group knowledge bases must be correct, consistent and consensual. This is in great contrast with what is currently developed over the networks: databases available through WAIS, Gopher or WWW do not have to be consensual nor consistent. The data is provided as such, without any warranty of consistency from the provider, and the retrievers have to make it consistent before introducing it into their own databases. Other frameworks are also different in this respect: neither the knowledge sharing (when the opportunity to modify the shared knowledge is considered) nor the software agents provide any warranty about the consistency of the knowledge they provide.

Moreover, the consistency is not only syntactic. The collaborative knowledge base building ensures that the collaborators agree on all of the details of the base and it is expected that they also agree on the meanings they assign to the terms. At the opposite, the knowledge sharing view allows the sharing of "ontologies" and mechanisms without caring that the meaning of the components are agreed (see also [10] about ambiguous names).

5.3.3. Good will and fair use

Of course, a system such as Co4 is not suited for just any purpose. It is designed for a community whose common interests lie in the results obtained by the community. In fact, Co4 is firstly dedicated to the members of large groups of people sequencing genomes in different laboratories for which the better the results of the community, the better those of the subscribers. Thus Co4 does assume from its users, their good will (they will submit their discovery and opposition to proposals) and a fair use (they will not delay some publication by not reviewing it, they will not consume more resources than necessary or submit proposals they know to be false). So, there is no place for financial refunding, quotas, deadlines or control on the queries emitted: it is assumed that the use of Co4 will be fair enough (no systematic submission of unverified material, etc.).

However, this does not mean that there is no conflict: the conflicts are to be treated by negotiation upon a protocol a bit stronger than those used in scientific journals because it is more formal and it aims at reaching consensus. Co4 includes an authentication of who proposed a modification and who issued an alternate proposal in such a way that the discoveries can be acknowledged. There is also the obligation of citing the sources in the hypertext annotations; this should be facilitated by the availability of a citation editor in Co4.

Co4 could be enhanced first by deadlines (for returning answers, etc. [26]) and second by punishments for those reviewers who take a very long time for reviewing (for instance, suspension of their submissions). There also could be, for reviewing, an anonymity management system which provides both independence and recognition [23].

Assuming good will and fair use from the agents of a distributed system is not unusual but not always explicitly stated.

5.3.4. The paper submission metaphor

Any system allowing the building of some artefact must have a particular change policy. The Co4 protocol mimics that of scientific journals. To our knowledge, the scientific journal protocol has never been used for that purpose. The choice of such a protocol is not neutral: first it is well known within the community and, in the consensual version, it enforces the dialogue between people (rather than a simple majority or intersection protocol). The requirements of consistency and formality allow for more strictness concerning what is published and thus leads to a consensual rather than a review protocol.

6. CONCLUSION

An architecture able to support the proposal of [19] has been presented. This proposal is original in two respects: it allows for the sharing of knowledge instead of non formalised data and it emphasises the consistency and the global coherency of shared knowledge. In summary, it is a formalised peer-reviewed scientific journal rather than a *dazibao*. For achieving this, we propose a *knowledge base* system able to deal with formalised and non formalised knowledge, *consistent*, because the formalised knowledge constitutes a consistent corpus, *consensual*, because whatever is deposited in the base is agreed by everyone, whose aim is the sharing of the knowledge base elaboration process.

Co4 has some limitations since the communication protocol is very restrictive (however, the same protocol does not prevent people from submitting to, reviewing for and reading scientific journals). However, it will have to be enhanced. The protocol itself is unable to account gracefully for the concurrent submission of mutually contradictory proposals. These major weaknesses are under consideration and will have to go through experimentation.

The framework presented here is not restricted to genome research but can be applied to other research fields, and to other activities such as the constitution of a corporate memory. Co4 is based on deep experience with individual workstation design and use. A complete prototypical knowledge base management system has been designed and implemented along these ideas. It has been used, in various stages of achievement, for the development of ColiGene and MultiMap, and is now being used for the implementation of a complete cooperative computer system for genome sequence analysis. The collaborative part is in its beginning: a prototype system is currently developed in our team using the knowledge base management system TROPES and KQML as a communication support.

ACKNOWLEDGEMENTS

This research is being supported by GREG (Groupement de Recherches et d'Études sur les Génomes) and by GdR CNRS «Informatique et Génomes» (CNRS: Centre National de la Recherche Scientifique). The author thanks Steve Jones, François Rechenmann and Jutta Willamowski for their help.

REFERENCES

1. Carlos Alchourrón, Peter Gärdenfors, David Makinson, On the logic of theory change: partial meet contraction and revision functions, *Journal of symbolic logic* 50(2):510-530, 1985

2. Chitta Baral, Sarit Kraus, Jack Minker, V. Subramanian, Combining knowledge bases

consisting in first order theories, *Computational intelligence* 8(1):45-71 1992

3. Laurence Cholvy, Robert Demolombe, Reasoning with information sources ordered by topics, Proc. 6th international conference on artificial intelligence: methodology, systems, applications, Sofia (BU), pp151-162, 1994

4. Clarence Ellis, Simon Gibbs, Gail Rein, Groupware — some issues and experiences, *Communication of the ACM* 34(1):38-58, 1991

5. Jérôme Euzenat, Building consensual knowledge bases: protocol, Internal report, INRIA Rhône-Alpes, Grenoble (FR), 1995

6. Tim Finin, Richard Fritzson, Donald MacKay, Robin MacEntire, KQML as an agent communication language, Technical report CS-94-02, University of Maryland, Baltimore (MD), 1994 (rep. in proc. 3rd CIKM, Gaithersburg (ML US), 1994) [ftp.cs.umbc.edu:/pub/ARPA/kqml/papers/cikm.ps]

7. Michael Genesereth, Steven Ketchpel, Software agents, *Communication of the ACM* 37(7):48-53, 1994

8. Sylvain Grivaud, François Rechenmann, Navigation dans les bases de connaissances associant objets et hypertextes, Actes 1er Représentation par objets, La Grande-Motte (FR), pp262-280, 1992

9. Thomas Gruber, Jay Tenenbaum, Jay Weber, Toward a knowledge medium for collaborative product development, in John Gero (ed.), Proc. 2nd. international conference on artificial intelligence in design, Pittsburg (PA US), pp413-432, 1992 [ksl.stanford.edu:/pub/knowledge-sharing/papers/shade.ps]

10. Peter Karp, Michael Mavrovouniotis, Representing, analyzing and synthesizing biochemical pathways, *IEEE Expert* 9(2):11-22, 1994

11. Rob Kling, Cooperation, coordination and control in computer supported cooperative work, *Communication of the ACM* 34(12):83-88, 1991

12. William Kornfeld, Carl Hewitt, The scientific community metaphor, *IEEE transactions on man, systems and cybernetics* 11(1):24-33 (rep. technical report AI-memo 641, MIT, Cambridge (MA US), 1981), 1981

13. Robert Kraut, Jolene Galegher, Carmen Egido, Relationship and tasks in scientific research collaboration, *Human-computer interaction* 3(1):31-58, 1987

14. Olga Mariño, François Rechenmann, Patrice Uvietta, Multiple perspectives and classification mechanism in Object-oriented Representation, Proc. 9th ECAI, Stockholm (SE), pp425-430, 1990

15. K. Narayanaswamy, Neil Goldman, “Lazy” consistency: a basis for cooperative software development, Proc. 3rd CSCW, Toronto (CA), pp257-264, 1992

16. Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, William Swartout, Enabling Technology for Knowledge Sharing, *AI Magazine* 12(3):36-56, 1991

17. Christian Overton, Kimberle Koile, Jon Pastor, GeneSys: a knowledge management system for molecular biology, in G. Bells, T. Marr (eds.), Computers and DNA, pp213-239, Addison-Wesley, Reading (MA US), 1990

18. Guy Perrière, Christian Gautier, ColiGene: object-centered representation for the study of *E. coli* gene expressivity by sequence analysis, *Biochimie* 75(5):415-422, 1993

19. François Rechenmann, Building and sharing large knowledge bases in molecular genetics, Proc. KB&KS workshop (International Conference on Building and Sharing of Very Large-Scale Knowledge Bases), Tokyo (JP), pp291-301, 1993

20. Walter Reinhard, Jean Schweitzer, Gerd Völksen, CSCW tools: concepts and architecture, *IEEE computer* 27(5):28-36, 1994

21. Reid Smith, The contract net protocol: high level communication and control in a distributed problem solver, *IEEE transactions on computers* 29(12):1104-1113 (rep. in Alan Bond, Les Gasser (eds.), Readings in distributed artificial intelligence, pp357-366, Morgan Kauffman, San Mateo (CA US), 1988), 1980

22. Mark Stefik, The next knowledge medium, *AI magazine* 7(1):34-46, 1986

23. David Stodolsky, Consensus journals: invitational journals based upon peer consensus, *Datalogiske Skrifter* 29, 1990

24. Hidetoshi Takana, A private knowledge base for molecular biological research, Technical report 811, ICOT, Tokyo (JP), 1992

25. Jutta Willamowski, François Chevenet, François Jean-Marie, A development shell for cooperative problem-solving environments, *Mathematics and computers in simulation* 36(4-6):361-379, 1994

26. Terry Winograd, A language/action perspective on the design of cooperative work, *Human-computer interaction* 3:3-30, 1987