SRC TR 89-55
UMIACS 89-76
CSD 2290

A Model for Computer Life

By

W. Litwin

A Model for Computer Life

Witold LITWIN
INRIA, 78150 Le Chesnay, France[1]

Nick ROUSSOPOULOS[2]
University of Maryland
College Park, MD 20742

Abstract

We propose a model to organize the logical level of the computers interconnected worldwide. We assume this universe populated with programs of a new kind called beings. Beings are autonomous, active and interoperable to the point they behave like living individuals. They are aware of their capabilities, able to use those of other beings, may self-modify and may give birth to new beings. They act autonomously, without our intervention. They may create organizations providing services beyond capabilities of a single being. We show the utility for the beings of a number of organizations we use for ourselves. Our expected profit from the computer life is less hassle in managing the computer universe, better operational capabilities at our disposal and financial profit from the lease of services of beings. We show that the model is basically feasible already and we point out the corresponding research issues.

Motto. *From the "Popular Journal", 1860 : Well informed people know that it is impossible to transmit voice over wires and that were it possible to do so, the things would be of no possible value.*

Historical note. *The telephone was invented in 1861. Shortly after, President of the Western Union Telegraph Comp. turned down exclusive rights to Bell's invention, explaining "What use could this company make of an electrical toy ?".*

---

[1] On leave from INRIA, with both Institute for Advanced Computer Studies and System Research Center.

[2] Also with the Computer Science Department, University of Maryland

1. INTRODUCTION

1.1. Evolution of data management goals

Computers will soon be interconnected worldwide. They will also carry systematically some database software. The idea in this software is the sharing of large data banks [Cod70]. The evolution of the technical developments was as follows :

- Data to be shared were removed from the original files and redefined as an integrated collection called database [Dat86]. The collection was under a central control of the database administrator, on a large mainframe.
- To improve performance, it was advocated to make the database distributed over a number of interconnected sites [Chu87], [Cer87]. The system was expected to provide the corresponding management.
- The data were allowed to be multiple autonomous databases. The system should provide functions for non-procedural manipulations of data in different databases [Lit87]. Data were assumed typically non-integrated, between the databases, as different administrators should have the autonomy of naming, choosing the data structures and values in priority according to their needs [Lit82], [Hei85], [Tem87], [Kuh88], [Jac88] , [Rus88], [Wol89].

The latter type of systems was called multidatabase systems, or federated systems [Lit82], [Hei85], [Day85], [Sar87]. Databases one may manipulate together without the integration were called interoperable [Lit86], [Int87]. Research showed that to provide interoperability, it may be useful for a multidatabase system to use autonomous services of other types of systems for data manipulation. For instance, to update dynamic attributes in the multidatabase system MRDSM, it revealed useful to call the formal calculus system MACSYMA. There was no question to integrate MACSYMA under MRDSM. The former system is several times larger than the latter [Lit87b].

1.2. Remaining drawbacks

This need leads to the idea of interoperable systems, which are autonomous systems that may be manipulated together without integration. Such systems are next step with respect to multidatabase systems [Lit89]. However, if designed according to the current principles, the interoperable systems will still leave a significant burden to the users. This burden will result from the following reasons :

- Computers carry a skyrocketing amount of software. The development of this software is now quite standardized and falls to well understood classes of skills such as : database systems [Bur86], text editors, compilers, image manipulators,... The programs realizing these functions duplicate a large number of

components realizing subfunctions or services [Gas87]. On the other hand each program lacks something useful for an application. Also, users are generally lost in subtle differences between various programs with similar purpose.

- Systems start now to be designed to provide the user autonomy [Abb88], [Gar88]. However, themselves, they are designed to do no more what the author and the user want and usually not less (unless they have bugs). They have no autonomy with respect to the designer or user and behave in predetermined way. If a new or unexpected function is needed, the only choice for the designer is to rewrite a part of the program. The user may either wait or try to find a program with the desired function which then may lack functions the user already had.
- A database person should also observe that there are systems that provide access to a very large number of databases, like the French videotex system TELETEL or the multidatabase system EasyNet. Nevertheless, the marketing staff of these systems talks less about databases and more about services. TELETEL provides in particular services like games, document translation, (pink) e-mail centers that may not use databases or involve the human behind the screen. The database approach considering the computer universe as a collection of data appears too limited.

1.3. A solution

All these problems mean that we need a model to organize the computer universe worldwide. We therefore propose such a model. It starts from the observation that the remarkable property of a life is that it continuously evolves towards a more efficient organization. Many efficient rules to organize the computer universe, are already embedded into our life. To set up a kind of a life mirroring ours inside the computer universe should thus be an efficient way to organize this universe as well.

To set up the computer life, we propose to populate the computer universe with programs of a new kind called beings. The beings are autonomous, active and interoperable to the point they behave like living creatures. They survive in general only if they generate profit covering their living expenses. They are aware of their capabilities and may use capabilities of other beings. They may self-modify and may give birth to new beings. They handle all the corresponding decisions autonomously ie without making us aware of. They may also create organizations providing services beyond capabilities of a single being. While these organizations are primarily for themselves, they mirror ours like banks, hiring agencies, schools, law enforcement institutions,...

It is argued that the computer life is a rewarding organization for ourselves as well. We should have less hassle in managing the computer universe. We will also use operational capabilities of beings that the computer life could create or improve by itself. Finally, we may gain financial profit from the lease of the services of the beings. The self-organization characterizing the computer life looks also anyhow the only way for the computer universe to reach the scale of millions of interconnected nodes we will require soon.

It is also argued that the model is not a science fiction. While almost not directly related work exist, the knowledge objects KNOs [TsI87] present some properties of beings, as well as the active objects with learning capabilities [McL88] and the actors [Hew86]. Also, the capabilities of beings called user agents, representing our interests inside the computer universe are these of mediators [Wie89]. More generally, the state-of-the art of the tools like multidatabase systems and database systems in general, object oriented paradigm and logic programming makes the model already basically feasible. We review the corresponding aspects of the model and show how these technologies let us to put the computer life into practice.

Section 2 discusses the model, its main concepts and rationales. Section 3 discusses specific details, resulting from the mapping of useful aspects of our own life. The discussion is centered on the feasibility of the model and on the research issues. Section 4 concludes the proposal.

2. THE MODEL

2.1. The basis

The model has for goal the organization of the computer universe at the logical level. Physical details, eg size of memories, type of computer used or interconnection procedures are irrelevant. The model starts from the observation that the main phenomenon organizing our universe is the life. Its remarkable property is that it is self-organizing and continuously evolving towards a more efficient organization. Many rules one may invent to organize a universe, are already embedded into our own life. To set up a kind of a life in the computer universe should therefore be an efficient way to organize this universe as well. This is the main axiom of the model.

A life is seen as an organization of a collection of individuals. The individuals seek for gains exceeding expenses for survival. They also let the collection to persist, despite the death of the individuals that could not survive. The individuals are autonomous and they interoperate. They also exhibit an active and adaptive behavior to survive. Finally, they are able to create new individuals to maintain their life forever.

The aspects of life pertinent to the model are these letting an individual to provide services to others. The gains in the computer life must therefore come from such activities. The purpose of the model is to be the methodology for organizing the computer universe towards this goal. The general approach is to map appropriate aspects of our life. Below, we outline the corresponding proposals. More details are in Section 3.

2.2. Beings

An individual inside the computer universe is called a *being*. A being is a new type of program. It is a tightly coupled, autonomous and active collection of capabilities oriented towards efficient service to other beings. The efficiency is measured by the difference between the gain from the services the being provides and its expenses for staying alive. The gains and expenses are measured by some kind of money. The capabilities themselves are programs that are not autonomous.

Formally, a being B is a triplet $B = (Co, V, Cs)$. Co are the *operational capabilities* of the being through which it provides the services. V (Curriculum Vitae) is a self-description of B, to make other beings aware of B's capabilities. Cs denotes the *survival capabilities* that allow B to survive and, especially, to find the way to earn more than it spends and to create new beings. They also allow B to autonomously decide to accept or refuse a task, to exchange information with other beings, to seek for employment etc. A being that cannot survive usually dies.

The birth of a being may result of an individual act as it may be collective, basically binary. The child may inherit some, but not necessarily all the capabilities of the parent(s). These aspects of the computer life are discussed later on.

The new crucial aspect of the autonomy of a being is that it is able to provide capabilities that were not built-in, to us and other beings. For this purpose, the being may apply external capabilities, like we use various tools tools. It is also able to make ad-hoc requests for service to other beings that may interoperate with it. Finally, it is able to import capabilities into itself for durable usage. Both Co and Cs sets of capabilities are thus time-dependent.

2.3. Organizations

Beings may form organizations providing more services than an individual being. An organization O is formally a quadruplet $O = (Co, V, Cs, \mathcal{B})$. $\mathcal{B}$ denotes the time-dependent set of beings forming O (employees of O). Any capability of O is that of an employee or is a composition of capabilities of it employees. An employee providing a service on the behalf of O is said to represent O. Employees may be in O indirectly ie through some O' nested into O.

The main difference between an organization and a single being is that the constituents of a being (capabilities) are not autonomous. An organization may also provides resources to the employees. Except for these differences, an organization basically gives to its customers the image of a large single being.

Organizations may also acquire capabilities that were not built-in and drop the useless ones. This is done through the evolution of the employees, but also through the hiring or firing of the employees. The evolution may be on the initiative of an the employee or may be requested by the organization. The corresponding decisions are autonomous with respect to us ie we are basically not aware of.

In particular, the computer universe will need a number of organizations useful for our own life. There should be hiring agencies, journals and universities for capabilities exchange and diffusion. There should be a kind of travel agencies for beings needing to move elsewhere. One will also need hospitals for damaged beings, courts, judges, police, etc. Not all beings will be gentle, as they may be set up by humans that are not [Cac89].

2.4. Autonomy, interoperability and competition

A being should be able to find operational capabilities it needs and does not have without our intervention. These capabilities are provided by other beings or organizations. The being should be able to choose them on competitive basis. It should also be able to sell its own capabilities (or of its organization) to other services (beings or organizations). The curricula vitae V are precisely for this purpose. Protocols for interoperability and operational capabilities should allow a being to exhibit its$V$ to others and to change it with the time. A being should also be able to to negotiate prices of services it provides or requests. In this sense, a being is aware of its existence, as we are aware of ours.

To provide a service, a being or an organization receive a request for. A request may be thought of as a high level message to an object or an actor [Hew86]. The properties of a being make however this concept richer than that those two, even that of an active objects [McL88]. Objects and actors are basically passive and present a much lower degree of autonomy and interoperability. These concepts do not require ability to self-change the set of capabilities, to give birth to other objects or to create organizations. The exception are the knowledge objects (KNOs [Tsi87]). The concept of a KNO is the closest to that of a being, remaining nevertheless more restrictive. Being are not required to communicate only through so-called *black boards* in [Tsi87], while a KNO does not carry a vitae etc.

2.5. Interface to our universe

Some beings will receive request for service from human users. They will provide these services by themselves or will issue requests to other beings. Such beings will be called *user agents* , extending the current meaning of this concept. A *marionette* KNO, is an example of a user agent. A user agent will however usually have capabilities of a *mediator* [Wie89], presenting thus a more autonomous behavior. A user agent may also belong to a user who is then its *owner*. The owner has the control of its agent, the priority for service and especially benefits from the financial gains from the services dispensed by the agent.

The degree of the autonomy foreseen for beings means that their behavior will be largely undeterministic with respect to this of the present programs. We will lose the control of the computer universe that will progressively rely upon itself. The self-organization created by the computer life is nevertheless probably the only practical way to allow the computer universe to reach the scale of millions of interconnected nodes we will need soon.

Our direct profit of the existence of beings should be two fold. We will dispose of their operational capabilities : to write a paper, to solve an equation, to schedule a travel,... The owner of a being may also to earn money for services of its being to others. This money may directly enter the owner's accounts in the computer universe that he may share with this of its agent. Alternatively, an agent may have its own accounting system, the owner's part being collected through some equivalent to our tax collection system. An owner may in fact have several agents and so while we speak about the owner as a human, this term may designate an organization as well.

It is also worthy to note that the owners will appear from the computer universe as a kind of Gods. There will be multiple Gods with autonomous conflicting interests and sometimes evil intentions. Fights between God's will become these of their agents and will influence the computer life. The human characteristics of Greek Gods seem a good framework for this level of the model (the human universe above the computer one).

3. FEASIBILITY OF THE MODEL

3.1. Overall analysis

While the model may appear as science fiction, it is already not. The services and of capabilities as the model considers them, may be described using the abstract data types or object oriented principles. Rules to be defined for various organizations to create, may be implemented using the logic programming. The production, evolution and the mutual understanding of curricula vitae between beings may be achieved through the self-description [Rou83], [Rou85], [Ccs87], [Bat88]. Efficiency of cooperation between distributed services may be achieved through the usage of techniques for parallel and distributed processing of autonomous data. They are now under intensive studies [Alo87], [Bre87], [Bel87], [Dee87], [Elm87], [Wie87], [Pu87], [Lit88].

The model adds to these paradigms the goals of autonomy and of interoperability at all levels. It also stresses the fundamental importance of these goals which only start to be fully appreciated [Eli87], [Dai88], [Gar88]. While, many technical problems remain to be solved, it is also promising that the model already worked rather well for ourselves.

To examine more in detail the feasibility of the model, we now review an example. We then focus on some concepts transposed from our own life that we show useful for the computer universe as well. We analyze their feasibility and we point out research issues.

3.2. Example

The user agent will usually reside on the user workstation. It is likely that the workstations will be individually owned and will be permanently a part of the computer universe connected, through the Open System Architecture, if necessary by radio [Hew85], [Iso87], [Dai88]. A workstation may be the location of the number of services and of beings, only some of them belonging to the workstation's owner. The composition of the services on the workstation will probably be decided by the user agent.

Consider that the user wishes to write some text. The agent may itself have the corresponding capabilities ie may be a *writer* (a sophisticated word processor in the current terminology). If not, it will call for a writer. The writer may be on the workstation or elsewhere. A distant writer may generate a clone that will travel to the workstation.

During the writing, it may happen that the user requests a capability that the writer does not have, such as to solve an equation. The writer attempts then to find this capability elsewhere. It either posts a request for a mathematician (equation solver in the current terminology) to delegate him the whole task or finds the capability somewhere and either applies it or even imports it into itself. In the latter case the capability may be dropped later on or kept permanently. The mathematician may travel as the whole or partly (as a clone) to the writer's workstation. Alternatively, the writer may send the equation to the mathematician or to the organization the mathematician works for.

To gain money, for its survival and/or for its owner, if any, the writer also seeks to sell its services to other beings. If idle, it is supposed to use the time to hunt for new capabilities to enhance its profitability. It may also advertise its services somewhere. It may finally do some housekeeping, update its vitae etc.

3.3. Implementation of beings

A practical kernel for the implementation of a being may be a dedicated multidatabase system (MBS), especially a relational one. A being will consist then from the system itself and from some databases under its management. The kernel and its databases will be identified by the multidatabase name that would be the logical name of the being itself. The commands of the multidatabase language of the system, like these of MSQL [Lit87] or of VIP-MDBS [Kuh88], will allow to merge in a single query data from different databases. They will also provide the interoperability with respect to data in the autonomous databases of other beings. Other features characteristic of an MBS or of a DBS in general, will allow for concurrent processing, for privacy, data security, etc. These features will be the kernel for the survival capabilities of the being.

A new fundamental feature of some of these databases should be the POSTGRES like ability to have attributes whose values would be the capabilities themselves [Sto86]. These capabilities will basically be the operational ones. Like in POSTGRES, a command, let us say EXECUTE, should then allow to

run the selected capabilities. A capability should be able to call another or even itself recursively. The flexibility of the multidatabase manipulation language of the system will allow to import and to export the capabilities. It will also allow to drop the useless ones. Finally, it will allow to define complex operations, involving several beings, through multidatabase transactions.

Other databases should contain data relative to the vitae. These data and the corresponding needs are discussed below. Finally, some databases may be necessary as the working environment for the capabilities.

Example. The writer could bear the logical name, let us say John Smith or John in short. Its kernel MBS may have a database named Op-Capabilities with the following relation :

Capab (Name, Type, Source, Version, Code)

where **Name** is the name of a capability, **Type** its type eg Speller, **Source** say where it comes from eg Microsoft, **Version** is the version id., and **Code** is the code to be executed. To check the spelling of a word, John may issue to its kernel the query :

```
USE    Op-Capabilities
EXECUTE Code FROM Capab WHERE Name = 'Speller'
```

Once in control, the speller will then ask John for the word to check.

A simple request for service to John from Nick could be :

```
USE    John . Op-Capabilities
EXECUTE Code FROM Capab WHERE Name = 'Speller'
```

To provide the service, John would allow the query to be executed by its MBS. IF John wants Nick to learn this capability, then John will issue the interdatabase query :

```
Use John . Op-Capabilities   Nick . Op-Capabilities
INSERT INTO Nick   . Op-Capabilities . Capab
SELECT *  FROM  John . Op-Capabilities . Capab
```

WHERE Name = 'Speller'

From now on, Nick will dispose of the capability by its own.

3.4. Curriculum Vitae

The Curriculum Vitae *V* is a fundamental tool for the interoperability of the beings. *V* should carry information allowing to evaluate its bearer skills. The evaluation should be performed autonomously, by the beings themselves, without our intervention. As for ourselves, a vitae should therefore contain the following parts.

3.4.1. Definition of the capabilities

Many programs have capabilities described today through the form of an explicit menu or of icons. However such a description usually rely on some implicit capabilities. These are not defined precisely or constitute implicit properties of the data model used eg database applications. The description may also rely upon hidden explicit capabilities in, so-called, Tool Boxes.

We therefore face the necessity of a language defining capabilities and their expression through other capabilities in some standard form. This description should form the *capability description* section of*V*. The description should be oriented towards the usage by the beings, not by the humans. Elementary language may consist of standard names of capabilities, eventually with parameters. Abstract data type approach and object oriented languages and systems are more elaborated candidates. A starting point may also be the capability description language like in [Rya86]. One should also consider the outcomes of the work on the self-description of databases [Rou82], [Rou83], [Rou85], [Mar87].

Anyhow, it does not seem practical to require the full description of each capability in each *V*. Therefore, one should consider some kind of external databases defining in depth capabilities. *V* should usually contain only the generic definitions, defining together the *profile* of the being.

Another problem is the evolution of a being. As it can gain new capabilities, it should be able also to add them to its profile. While the general solution is an open issue, it is nevertheless rather easy to see how to manage simple profiles through an MBS.

3.4.2. References

The process of evaluation of capabilities should also be autonomous. A helpful tool may be the *reference* section in *V*. This section should describe the work experience of the being. If the being is often used for instance, then it is likely to be a better choice than the other usually idle. One may also collect names of beings to refer to for the opinion about the previous work. One issue is again the language for the description of the corresponding information. Again, while the general solution is an open issue, it is easy to see that a lot may be achieved if the kernel of the being is an MBS.

3.4.3. Salary

Any being is supposed to generate some profit. The vitae *V* should therefore indicate the price of the services of the being, in some kind of a *pricing* section. Again, the pricing should be established basically by the being itself. Since the pricing strategy may be complex, an interesting issue is how it should be done.

Again, external databases may be helpful to define the profitability ratio, given the profile of the being. This problem is addressed more below, in the section dealing with the survivability.

3.4.4. Interview

The generic capabilities announced in *V* may be not understandable enough to another being. Also, the confidence to *V* should be limited. Therefore, the being examining *V* of another one should be able to interview its bearer. An interview may consist of request for details of capabilities. It may also include the benchmarking of corresponding performance. It may be done by a specialized service.

3.5. Organizations

The beings that formed an organization should be able to pool their capabilities for the group work. This leads to a number of technical issues. We outline a few:

- joint representation as an organization. An organization may need in particular to be nested.
- choice of the most efficient manner to perform the work : either information may move to the being or vice versa. In the latter case it may be useful to give birth to an offspring (limb [Tsi87) tailored to the task.
- internal structure of the organization and its its conceptual scheme.
- internal communication and management of various dependencies [Mar85].
- some kind of legal responsibility, with respect to the overall privacy and security rules of the universe.

Again the proposed way to implement the beings may obviously help to achieve these needs. The negotiation protocols may be an extension of those proposed for federated databases in [Hei85].

3.6. Exchange of capabilities

It is assumed that some beings are able to export capabilities. In general a being is also able to use external capabilities. One may foresee the following ways of doing it :

(a) - the being calls another being or organization. This being or a representative of the organization executes then the task requiring the capability on the behalf of the requestor. This is the groupware approach.
(b) - the being learns through some script how to use capabilities available in some recipients eg database or knowledge base. These capabilities remain outside the being. It only uses them as we use tools helping us to do a task.
(c) - the being learns the capability which means that it imports the corresponding code into itself.

An elementary possibility to realize (a) may be today for instance the technique of the remote procedure call. This means that the requester knows who provides the service. If this information is unknown, one may design a kind of broker server describing who provides a given service, how much it costs etc. Services may then be searched for using database or knowledgebase queries.

This type of cooperation will require also standards for the interfaces between capabilities. Many such standards exist de facto already, especially for microcomputer software. There are many independently developed auxiliary programs in this area, able to interoperate over data in format of some main programs. For instance, there are several speller checkers that may be called to operate on the text of MsWord while it is being written, as if they were parts of the MsWord.

In the case (b), in general, the being will need to purchase the right to use capability from some server. It may then get a script (transaction scheme) to be downloaded it into its private database. Like a manual, the script will define how to use together local and server's capabilities, what data should be sent to the server etc. To use the server may be advantageous, cheaper, more efficient if the server is on a powerful computer, and more up to date.

However, the strategy (c), ie the self-import of a capability may lead to similar advantages. It also offers a better control over the capability. The implementation of this strategy may require nevertheless software engineering techniques, we are not aware of presently. However, while one may foresee technical problems related to the transition between modified source code, new object code, linking etc., none of them seems really hard. One basis for (c) may be techniques for extendible database systems, provided their are generalized to make an extendible system acting autonomously.

On the other hand, the use of an MBS as a kernel, also may allow to realize the goals (a) to (c)., as the following example shows.

Example. Assume that CompuServe server has the database **Capabilities** with the following scheme :

```
Capab (Name, Version, Code, Interface, Type)
Cap_pricing (Name, Rent, Buy&Share, Buy&Copy, Update)
Usage (User-Id, Name)
Volume_Discount (User-Id, Name, Discount).
```

To download cheap equation solvers, the writer John may issue the following MSQL query with obvious meaning:

```
USE CompuServe.Capabilities John.Op-Capabilities
INSERT INTO Op-Capabilities.Capab
LET X BE CompuServe.Capabilities
SELECT X.Capab.Name X.Capab.Version X.Capab.Code X.Capab.Type
FROM X John.Op-Capabilities
WHERE X.Cap_pricing.Buy&Copy < 10 $ AND X.Capab.Type = 'Equation solver'
AND X.Capab.Name = X.Cap_pricing.Name
```

3.7. Life cycle of a being

3.7.1. Birth

Three basic mechanism for birth appear useful, from the analysis of our universe.

3.7.1.1. Offsprings

When some beings evolved a lot, or they should pool their capabilities, it may happen that the most efficient solution is to produce an offspring merging the required capabilities and free from some inefficiencies due to the aging of the internal structure of its parents. We do not know current technical solution to this problem, but these that come to mind at the first glance do not seem very complicated (eg production of a new join source code and recompilation or a duplication of the MBS with an appropriate selection of the capabilities and of the related data). It also leads to the problem of some certification of new beings, to avoid the trouble once their entered the life.

3.7.1.2. Cloning

Cloning is the production of a being by a single being, for instance to perform some remote task. The clone is a copy of its parent with some of its operational capabilities. Viruses are examples of multiplication by cloning, as well as limbs in [Tsi87].

3.7.1.3. Act of God

This is the usual way to create a program today. The program is entirely created by the human user. This type of birth should progressively disappear. One may rather envisage more subtle acts, providing a clone or an offspring with capabilities none of the parents had.

3.7.2. Survival of a being

It is fundamental for the computer life that only useful beings remain in the universe. This leads to the problem of the corresponding criteria. The profitability appears the best one. The corresponding principles may then be as follows :

- there is an overall notion of money
- any service provided is payable,
- usage of any shared resource is a service to be paid, including the occupation of some storage for the being itself.
- any being must be able to pay its running expenses.
- these expenses may include some transfer of money to a God account, in a similar way we pay taxes.

For this purpose the being must gain money, through services to other beings. An interesting technical problem is how the money circuit should be organized itself and whether one should have cash money as well, or accounts only. Another interesting problem are the strategies to keep the being's balance at least positive or to make it highest possible.

3.7.3. Illness

Occasionally it may happen that a being gets damaged, for instance because of software or hardware element of the universe failure. While the design allowing for some autorecovery is feasible, especially through MBSs, it is clear that specialized services will be needed. One may call them of course doctors and hospitals inside the computer universe. We are not aware of general techniques for the corresponding medicine.

3.7.4. Death

The non-profitability (a definitely negative balance) may be the criterium for the removal of a being from the universe, ie its death, although not the only criterium. There is the technical problem of the design of the death. Our own organization suggests that in general the death in the computer life can not be a simple removal. It will need to involve some legal aspects : closing of the associated accounts, sending of messages around etc. Some corresponding issues are already discussed for KNOs in [Tsi87].

3.8. Law enforcement

As for ourselves, some laws will be necessary for the beings. Initial laws should be set up by us, one may call them by analogy Law Tables or Commandments. On this basis, the laws of the universe should then evolve autonomously. It remains an open question how the laws should be coded and how one should organize their evolution. Again, it results from the model that we should take the corresponding organization of our universe as the submodel for this purpose.

It is also clear that there will be malign beings performing operations against the privacy and the security of others. Such attempts already happened and are dangerous even if the programs are not beings simply worms [Cac89]. The techniques already known for databases will provide some protection. However, law enforcement institution will be necessary also for the computer life, if it should be autonomous. This leads to the problems of the law encoding, of the access to the dedicated knowledgebases and of the organization of law enforcement institutions. It is an open question to what extend the enforcement may be made preventive or even in the real time.

One approach may be to use tools and provisions of a system like ITOSS [Rab89]. Some beings should then be the "security engineers", in charge of the security policy. In general, the concepts like sentinels, secure committee, or fingerprints in ITOSS seem particularly appropriate for the autonomous building of the security of the computer life.

4. CONCLUSION

The principles of autonomy and of interoperability lead to the organization of the computer universe as a life modeling our own. Living creatures are then a new type of programs called beings that are more general and autonomous than objects or actors. They seek to survive, may import and export capabilities, may multiply and may exhibit to other beings the description of their skills and of their experience. They are also able to create organizations. We have shown through our model that a number of organizations we have created for ourselves will need to be transposed to the computer universe as well.

By the nature of life itself and its embedded search of effectiveness, the model appears attractive. We have shown its foundations and proposed a framework for the operational setup. We conclude that the computer life appears feasible and useful, although its creation is a major enterprise. Still, its basic features may be implemented surprisingly easily. Multidatabase systems extended with POSTGRES like features to execute capabilities stored in relations appear a preferential tool for this purpose.

While we have surveyed a number of technical issues related to the viability of the model, we did not enter into details like in a typical research contribution. It is nonetheless a fundamental role of researchers to propose sometimes a new general framework as well. The specific technical contribution find then there their overall purpose. This is now the time for these proposing operational solutions for the implementation of the computer life. As the model shows, the corresponding perspectives are vast and fascinating.

Acknowledgments

We would like to thank Dennis McLeod and Gio Wiederhold for helpful suggestions.

This work was supported by the Institute of Advanced Computer Studies of the University of Maryland (UMIACS), by the National Science Foundation under Grant CDR-85-00108 and by the Institut National de Recherche en Informatique et en Automatique (INRIA) under the INRIA - SRC cooperation protocol.

REFERENCES

[Abb88] Abbott, K. R., McCarthy, D. R. Administration and Autonomy in A Replication-Transparent Distributed DBMS. 14-th Int. Conf. on Very Large Databases, Los Angeles, USA, (Aug. 1988), 195 - 205.

[Alo87] Alonso, R., Garcia-Molina, H., Salem, K. Concurrency Control and Recovery for Global Procedures in Federated Database Systems. IEEE Data Engineering, (Sep. 1987), 10, 3, 5-11.

[Bat88] Batini, C. Di Battista, G. A methodology for conceptual documentation and maintenance. Inf. Syst. 13, 3, 1988, 297-318.

[Bel87] Bellcastro, E & all. DQS -Distributed Query System. (Sept. 1987), CRAI, Italy, 21. Int. Conf. on Extending Database Technology, Springer Verlag, 1988.

[Bre87] Breitbart, Y., Silberschatz, A., Thompson, G. An Update Mechanism for Multidatabase Systems. IEEE Data Engineering, (Sep. 1987), 10, 3, 12-18.

[Bur86] Burns, T. Fong, E. Jefferson, D. Knox, R. Reedy, C Reich, L. Roussopoulos, N. Truszkowski, W. Reference Model for DBMS Standardization. ACM SIGMOD Records, (March 1986).

[Cac89] Special Section on Internet Worm. CACM, 32, 6 (June 1989), 677-710.

[Cod70] Codd, E., F. A Relational Model of Data for Large Shared Data Banks. CACM, 13, 6, 1970, 377-387.

[Ccs87] Consultative Committee for Space Data Syst. : Standard Formatted Data Units - Structure and Construction Rules. Red Book, Issue 2, (Feb. 1987). Nat. Aeronautics and Space Adm.

[Cer87] Ceri, S., Pernici, B., Wiederhold, G. Distributed Database Design Methodologies. Proceedings of the IEEE, (May 1987), 533-546.

[Chu87] Special Issue on Distributed Database Systems. Chu, W. (ed). Proceedings of the IEEE, (May 1987), 532-735.

[Cze87] Czejdo, B., Rusinkiewicz, M., Embley, D. A Unified Approach to Schema Integration and Query Formulation in Federated Databases. Res. Rep. University of Houston, 1987, 25.

[Dai88] Distributed Aspects of Information Systems (DAISY Working Group Rep). Research into Networks and Distributed Applications. R. Speth (ed.). Elsevier Science Publ. 1988, 1029 - 1049.

[Dat84] Date, C., J. A Critique of the SQL Database Language. SIGMOD Record, 1984, 8-54.

[Dat86] Date, C., J. An Introduction to Database Systems. 4-th Ed. Vol. 1. Addison-Wesley, 1986, 639.

[Day85] Dayal, U. Query Processing in a Multidatabase System. Query Processing in Database Systems, 1985, Springer Verlag, 81 - 108.

[Dee87] Deen, M., S.,Amin, R., R. Taylor, M., C. Data Integration in Distributed Databases. IEEE Trans. on Soft. Eng., 13, 7, (July 1987), 860-864.

[Eli87] Eliassen, F., Veijalainen, J. Language Support of Multi-database Transactions in a Cooperative, Autonomous Environment. IEEE Region 10 Conf., Seoul, (Aug. 1987).

[Elm87] Elmagarmid, A., Leu, Y. An Optimistic Concurrency Control Algorithm for Heterogeneous Distributed Database Systems. IEEE Data Engineering, (Sep. 1987), 10, 3, 26-32.

[Fan88] Fankhauser, P., Litwin, W., Neuhold, E., Schrefl, M. Global View Definition and Multidatabase Languages : Two Approaches to Database Integration. Research into Networks and Distributed Applications. R. Speth (ed.). Elsevier Science Publ. 1988, 1069-1082.

[Gar88] Garcia Molina, H., Kogan, B. Node Autonomy in Distributed Systems. IEEE Int. Symp. on Databases in Parallel and Distr. Systems. 1988, 158-166.

[Gas87] Gash, B., Kelter, U., Kopfer, H., Weber, H. Reference Model for the Integration of Tools in the "EUREKA Software Factory". ACM-IEEE Fall Joint Comp. Conf. (Oct. 1987), 183-190.

[Ham79] Hammer, M., McLeod, D. On database management system architecture. MIT Lab. for Comp. Sc. MIT/LCS/TM-141, (Oct 1979), 35.

[Hei85] Heimbigner, D., McLeod, D. A Federated Architecture for Information Management. ACM Trans. on Office Information Systems. (July 1985), 3, 3, 253-278.

[Hei87] Heimbigner, D. A Federated System for Software Management. IEEE Data Engineering, (Sep.1987), 10, 3, 39-45.

[Hew85] Hewitt, C., De Jong, P. Open Systems. On Conceptual Modeling. Springer Verlag, 1985, 147-164.

[Int87] Interoperable Database System. 1st International Symposium. INTAP, (May 1987), 167.

[Iso87] Remote Database Access Protocol. 2-nd Working Draft. ISO/TC 97/SC 21/WG 3, 1987.

[Jac88] Jakobson, G., Piatetsky-Shapiro , G. Lafond, C., Rajinikanth, M., Hernandez, J. CALIDA : A Knowledge-Based System for Integrating Multiple Heterogeneous Databases. 3-rd Int. Conf. on Data and Knowledge Bases : improving usability and responsiveness. Jerusalem, (June 1988), Morgan Kaufmann Publ., 3-18.

[Kuh88] Kuhn, E., Ludwig. Th. VIP-MDBS : A Logic Multidatabase System. IEEE Int. Symp. on Databases in Parallel and Distr. Systems. 1988, 190-207.

[Li87] Li, Q., McLeod, D. Object Flavor Evolution through Learning in an Object-Oriented Database System. 2nd Int. Conf. on Expert Database Systems. The Benjamin/Cummings Publ. Comp. 469-495.

[Lit82] Litwin W. et al. SIRIUS Systems for Distributed Data Management. Ed. H. J. Schneider. North-Holland, 1982, 311-366.

[Lit86] Litwin W., Abdellatif, A. Multidatabase Interoperability. IEEE Computer, (Dec. 1986), 19, 12, 10-18.

[Lit87] Litwin W., et al. MSQL : a Multidatabase Language. INRIA Res. Rep. 695, (June 1987), 41. To appear in Inf. Science - An International Journal, Special Issue on Databases, 48, 2 (July 1989).

[Lit87b] Litwin W., Vigier, Ph. New Functions for Dynamic Attributes in the Multidatabase System MRDSM. HLSUA Forum XLV, New Orleans, (Oct. 1987), 467-475.

[Lit88] Litwin W., Tirri. H. Flexible Concurrency Control using Value Dates. IEEE Distr. Proc. Techn. Comm. Newsletter. Spec. Issue on Heterogeneous Distributed Database Systems, 10, 2, Nov, 1988, 42-49.

[Lit89] Litwin W., Mark, L. Roussopoulos, N. Interoperability of Multiple Autonomous Databases. System Research Center. Univ. of Maryland, College Park. Techn. Rep. TR 89-12, 45.

[Mar85] Mark, L., Roussopoulos, N., Chu, B., Update Dependencies. IFIP TC2 WG 2.6 Working Conference on Database Semantics, (Jan. 1985), Belgium.

[Mar87] Mark, L. Roussopoulos, N. Information Interchange between Self-Describing Databases. IEEE Data Engineering, (Sep. 1987), 10, 3, 46-52.

[Mar87a] Mark, L., Roussopoulos, N. Operational Specifications of Update Dependencies. SRC Res. Rep., Univ. of Maryland, (Feb. 1987), 44.

[McL88] McLeod, D. A Learning-Based Approach to Meta-Data Evolution in an Object-Oriented Database. Advances in Object-Oriented Database Systems. Springer-Verlag Lecture Notes in Comp. Sc., 1988. 219-224.

[Mot87] Motro, A. Superviews : Virtual Integration of Multiple Databases. IEEE Trans. on Soft. Eng., 13, 7, (July 1987), 785-798.

[Neu88] Neuhold, E., Schrefl, M. Dynamic Derivation of Personalized Views. 14-th Int. Conf. on Very Large Databases, Los Angeles, USA, (Aug. 1988), 183-194.

[Pu87] Pu, C. Superdatabases : Transactions Across Database Boundaries. IEEE Data Engineering, (Sep. 1987), 10, 3, 19-25.

[Rab89] Rabin, M., Tygar, J. D. ITOSS: An Integrated Toolkit for Operating System Security. Foundations of Data Organization and Algorithms. W. Litwin & H.-J. Schek (Eds). Springer-Verlag, 1989. 2-15.

[Rou82] Roussopoulos, N. Wallace, S. Self-Description vs. Self-Documentation. Workshop on Self- Describing Data Structures, Univ. of Maryland, October 1982.

[Rou83] Roussopoulos, N. Intensional Semantics of a Self-Documenting Relational Model. Dept. of Computer Science, Techn. Rep. 1264, (April 1983), Univ. of Maryland.

[Rou84] Roussopoulos, N. Mark, L. Update Dependencies in Relational Databases. 1st International Conference on Expert Database Systems, Kiawah Island, South Carolina, (Oct. 1984).

[Rou85] Roussopoulos, N. Mark, L. Schema Manipulation in Self-Describing and Self-Documenting Data Models. Int. J. of Comp. and Inf. Sc., 14, 1, 1985, 1-28.

[Rus88] Rusinkiewicz, M et al. Query Processing in OMNIBASE : a loosely coupled multi-database System. Tech. Rep. #UH-CS-88-05, Univ. of Houston, (Feb. 1988), 27.

[Rya86] Ryan, K., Larson, J. The use of E-R models in capability schemes. 5-th Conf on E-R Approach. Dijon, France (Nov. 1986).

[Sam88] Samy Gamal-Eldin, M., Thomas, G. Elmasri, R. Integrating Relational Databases with Support for Updates. IEEE Int. Symp. on Databases in Parallel and Distr. Systems. 1988, 202 - 209.

[Sar87] Special Issue on Federated Database Systems. Sarin, S. (ed.). IEEE Data Engineering, (Sep. 1987), 10, 3, 64.

[Sto86] Stonebraker, M. Rowe, A. L. The Design of POSTGRES. ACM-SIGMOD 86, 340-355.

[Tem87] Templeton, M. et al. Mermaid : A Front-End to Distributed Heterogeneous Databases. Proceedings of the IEEE, (May 1987), 695-708.

[Tsi87] Tsichritzis, D. Fiume, E., Gibbs, S., Nierstrasz, O. KNOs: KNowledge Acquisition, Dissemination, and Manipulation Objects. ACM-TOIS, 5, 1, (Jan 1987), 96-112.

[Wie87] Wiederhold, G., XiaoLei, Q. Modeling Asynchrony in Distributed Databases. 3rd IEEE Conf. on Data Engineering, Los Angeles, (March 1987), 246-250.

[Wie89] Wiederhold, G. The architecture of Future Information Systems. Draft. Stanford University, (Jan. 1989), 17.

[Wol89] Wolski, A. LINDA : A System for Loosely Integrated Databases. 5-th IEEE Conf. on Data Engineering, Los Angeles (Feb 1989).