Proceedings of the 6th International Workshop on Plan 9, October 20-21, 2011, pp. 1-10
Universidad rey Juan Carlos, Madrid, Spain. ISBN 978-84-694-9010-5
(With a few corrections.)

From natural hazards to outer space and to Plan 9

Vincent Douzal, Cemagref, UMR Tétis, Montpellier
vincent.douzal@teledetection.fr
Nicolas Bercher
nbercher@yahoo.fr
David du Colombier
djc@9grid.fr
build date: 2012-03-01 09:16

ABSTRACT

An information system for scholarly work on natural hazards calls for the design of a computer system for transmission of information over very long periods, and for traceability. An abstract analysis shows that these requirements are dual to the fundamental question of assisting the cognitive activity of a user using external memories, which reaches a very general scope. The solutions should be implemented at the operating system level, mainly the file system, and Plan 9's file systems and other properties make it the soundest base for our work. We present our road-map for development.

1. Introduction: the case of a computer system for scholarship on natural hazards

We were requested to design an information system supporting scholarly work on natural hazards. It quickly appeared that all visible, mainstream systems sorely lacked some core properties, thus incurably disqualifying them, notably in their relation to time. We therefore considered a more fundamental problem: building the necessary foundations of a system to work on natural hazards. Early on, it became obvious that many requirements for the system were not specific to that field. Even so, 'natural hazards' provides a good concrete example to guide the design. For example, it naturally involves century-long durations, which has significant implications for the design. Furthermore, by their nature, the natural events that will form the data records are unpredictable, and their nature often precludes making precise observations at the time. Crucial data arrives unformatted from endangered persons. Later, historians will delve into archives, sifting centuries-old files, seeking to reconstruct events. That is exactly the process we want to assist. The context of the reconstruction or investigation matters too. For example, if the work is contractual it might be brought before a judge for arbitration, which would proceed by retracing its sources; similarly, a scientist or scholar might need to retrace arguments made earlier by others, and check results. On behalf of the earlier author of any conclusion, one should be able, provably, to revert to the exact state of information he had recorded at a given date, and perform again the same processing steps. The key concept, then, is traceability. The crucial need for facts is not bound to a present, limited state of science: more elaborate models (physical simulations, for instance) always require more facts as limiting conditions. Thus, data collected today might still be required in a remote future. For working on natural hazards we see a need to design an information system for the next 500 years! That leads us to the question: what is invariant over such time scales?

The ''sciences of traces'' — history, archaeology, paleontology, geology, cosmology (on ''memories of the world about itself''), the now fashionable forensic medicine — all share the essential structure of a police investigation. This hints that, when thinking on scales of

centuries, any design will be totally unspecific to any particular application domain. Traceability is the key element to reproducibility in science. Let us establish that two questions are dual in the mathematical sense (they are two sides of the same coin): 1. The very long-term transmission of knowledge established on digital memories. 2. The absolutely minimal functionalities a computer should fundamentally provide for sustaining, assisting a user in his cognitive activity.

All the usual assumptions for designing an information system are swept aside by duration. Space coordinates are taken as an invariant index in geographical information systems, but space shrinks or expands during seismic events. Continents drift. Concepts evolve. Languages change: a document in Old English is in a foreign language. No application ''ontology'' could be expected to last, only very abstract concepts. One cannot envisage a data model (in the usual sense) for a very long-term repository. Hence a very long-term database should be completely agnostic to the kind of data it will store, and have no data model at all.

2. Into outer space: An abstract analysis, for the essentials of the structure of an external memory system

Our model of the system can be expressed visually by Figure 1. The α part depicts our expanding universe unfolded in space-time, in a way standard in physics since Minkowski. The world at a given point in time τ is a slice of space-time, represented as a Venn diagram: the set of perceptible events at τ. By construction, events are fixed points: unchanging, invariant entities. The arrow of time τ is irreversible.

Two notable kinds of events are shown: those leaving long-lasting tracks (a fossil, a footprint) appear like tubes, and those which don't (like a perfume spray). Observing a slice of a tube implies an event in retrospect, if only one has the ability to interpret tracks, and traces. Otherwise past events are lost.

If we are to capture facts about events (in ω), we need to take their image under a given relation, in the sense of set theory. An image in the form of a series of symbols is the convenient way to put things in writing, into an external memory. This relation from events to categories and eventually symbols takes varied forms, some are termed perception, modelling, measurement, or even database form filling. Recordings of these assignments come in sequence along a time line.

The α and ω parts display an inviting symmetry. Since events are fixed things, so should be their images: written once-for-all, never altered again. Images, if they are to play their rôle, must be written on lasting, immutable external memories—tubes. External memories act as messages; they offer to parties of a contract the cross-checking abilities that are essential to their use, like the physical tokens of Neolithic accounting systems [16].

The whole purpose of the diagram's ω is to break symmetry with α on time; unlike τ, t is reversible: it can be travelled up and down. The impossible motion back and forth on τ is replaced by a motion in space, between preserving tubes of memory indexed by t.

Together with *t*, a user ID *uid*, and a unique repository name *host*, form the triplet ID of a record. These are the only metadata elements guaranteed to appear for every record. (Obviously, they are strictly internal to the recording system.)

Note that the machine representation for t does not have to start at the (hypothetical) origin of τ for the universe: an artificial epoch is enough, such as 1970 in Unix time, or the more recent date of establishment of the repository. The stated dates of events are to be written in the records themselves, using for τ a calendar representation, to which machine time must naturally be connected (with that connection deposited periodically in records). Pointing to cosmic, geological, paleontological remote events on τ is done by ''inductive referral'', through a theoretical construct. For example, consider the chain of hypotheses and processes involved in carbon-14 dating an artifact as 14,000±500 years before present, which in effect builds a virtual time arrow. Similarly, description of where events take

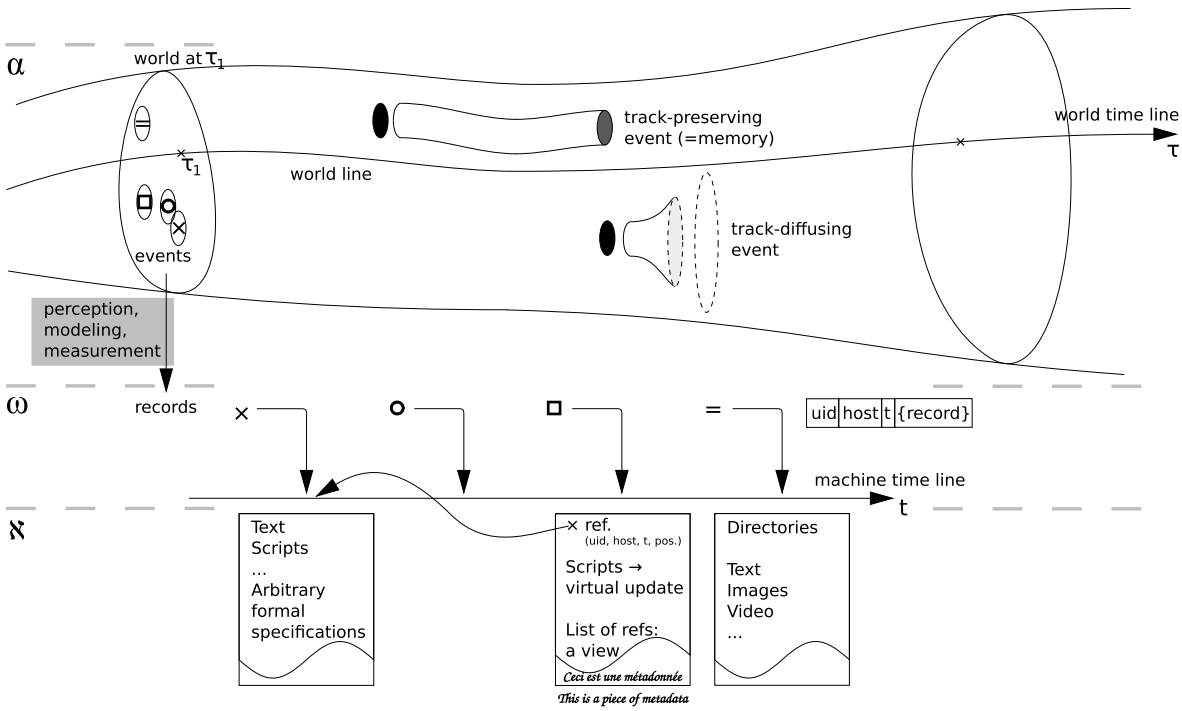

Figure 1. The whole article can be considered an extended legend to this diagram for the abstract setting of our design. The idea of including α *in the form of a Minkowskian diagram was inspired by Schueler's remarkable paper [17].*

place is not primary, but secondary; it is of course of importance, but cannot be enforced in the records.

Machine time t is the only organising dimension of the repository. It is a simple order, expressing the intrinsic 'sequentiality' of any subjective experience. Sequentiality is the only structure inherent to the system by design. It is the only indispensable and unavoidable structure that should possess an external memory system, as reflected by the diagram; and no other structure should be put in without harm.

An elementary cognitive operation consists in picking two arbitrary memories, experienced at any times, and relating them in some manner. Thereby for instance an insignificant event in childhood suddenly makes sense in the light of a recent reading. Relating two events is done in א by referring to their records. In an א record, referring to a past record is to open a *view* on it: a generalisation of what a directory provides on usual file systems. A view can gather arbitrary records, comment them freely, support action on (parts of) its constituent files through scripts, and include visualisation and user interaction facilities. A view could also be used for instance to provide another user with visibility and rights of accessing a given set of files, or to display several parts of files as a single body. We see here the demand for a computer language to express such views.

Diagrams ω and א represent well the progress of, say, an historian's trajectory through an archival corpus, making records, reconsidering them, viewing them arranged in any order he likes. More broadly, they represent an investigation in any ''science of traces'', and even more generally, the acquisition of knowledge along a lifeline, in a linear succession of

encounters of people, situations, books, etc., all marking events in time. At an abstract level all these processes are formally identical: ω and א together capture some essentials of cognitive activity and of an external memory system for expanding, augmenting cognition. The corresponding basic information system architecture is that of a computerised log book. A ship's log book is a perfect characterisation of a lifeline, it reflects in its sequential records the ship's navigated trajectory around the sea, and its conjunction with significant events at each time and place. So much that metonymy had both named *periplus.*

Our design minimally implies two components, a linear, sequential file system, strictly 'write once', and a language to express free comments, references to records, and invoke arbitrary scripts on records.

3. Long-haul message passing into the future: the need for a corpus

Any writing is a message sent into the future (if only to oneself). A message makes sense because of a convention, a contract between parties. If my collaborator voices ''42'' over an experimental equipment, that can be a perfectly qualified piece of data between us. On the other hand an isolated file containing ''42'' makes sense to no one. Agreeing on a given protocol allows a message to achieve its intended purpose, carrying meaning, and is called interoperability. Applications agreeing on a file format achieve so-called technical interoperability. Conforming to shared ''ontologies'' allows so-called semantic interoperability. By contrast, messages between humans have no static semantics; they must be submitted to interpretation, just as any track requires the skill to interpret it to reveal a past event. Interpretation needs to know the surrounding context of the message, a larger context with cultural distance.

For data, that context is a *corpus*: a substantial body of information with internal closure and additional connections to our present world, enough that we can interpret a given item. The further into the future a message is sent, the more it needs a whole corpus of documents as a context to convey sense, because interpreting the message implies reconstruction of the whole sender-receiver-transaction-motivation setting. Protocols and conventions, and cultural values evolve. Scripts get forgotten, and only when a substantial corpus of connections, with enough closure is available can they be deciphered, as with Egyptians hieroglyphs or the Maya script. It follows that a corpus is the best basis of trust. Internal closure eases forgery detection. Traceability is a fair basis for reputation.

The DOI system documentation [3] asserts that managing data implies managing relationships between entities: A has the relationship B to C. For instance, ''Albert Camus is the writer of La Chute'', an expression that directly maps into, say, RDF. Provenance should not be omitted: who said that, and when. Relationships between entities is often called metadata.

A corpus is ''extreme metadata''. Clearly any record in our repository is metadata to any other one it references, and conversely. Even more so, any set of contemporary records are implicitly but strongly connected just by being close in time: a Freudian slip, or even an empty record, can convey meaning just from its position after an otherwise unrelated record.

4. Contrasting with current architectures: enforced order

All current systems force data into a Procrustean bed of arbitrary orders, ultimately distorting it to the point of defeating its use, and even obstructing its entering the data system.

Current file systems insist we arrange files in a hierarchy. Thus we try to express semantics in a hierarchy. How long can a hierarchy, however carefully devised, accept new subjects without breaking? Also, much too often, a file should legitimately be in two different places. Large libraries confront the same dilemma [18], when two yet unrelated disciplines, remote in classification and physical arrangement give birth to a completely new

one. (In a small collection these problems are imperceptible, because an exhaustive scan makes up for orderliness.) The negative consequences of imposing arbitrary order on data where it does not belong poisonously pervade every task. A file system hierarchy also cannot grow indefinitely without requiring foreign mechanisms, such as logical volumes. By contrast, our linear sequence of records naturally splits up in articles and extends easily by adding memory devices. For the same reasons, it lends itself well to parallelism.

Relational databases similarly force data into the relations of a given data model. Diagram ω shows how a very long-term database can be agnostic to the kind of data it will store, by reducing to a single table with key *uid, host, t*. Hence no database system is needed, just a file system. (If you do want a database, however, there is no difficulty in periodically storing its current state and log inside a record.) A database model somewhat supposes it exhausts the possibilities of what will have to be recorded in the future.

Forcing inappropriate order, current systems make it difficult to accommodate really new concepts. What protocols, what conventions will be demanded in the future, we cannot foresee and must not prejudge.

The computer system outlined in the diagram — a computer being a memory with an engine to process data — retains only the minimum necessary to support cognitive activity, and allows transmission of knowledge into the distant future.

5. Storing files as the privileged digital abstraction

What digital abstraction should we make accessible to the user? Files are almost universally available, well understood, and uniformly usable: they hold everything. We feel we also cannot avoid file hierarchies, because most systems present themselves in that form and can only be used in that way. Accepting hierarchies does *not* mean forcing everything into hierarchies: it is only a facility offered locally in a record.

The intended linear, sequential file system uses Venti [10] as block storage layer. Unicode is the encoding of choice, with its universal scope and strict stability principle, consistent with our concern.

6. Correcting errors without hard update

Requiring traceability has a drastic consequence: since updating contradicts traceability, in our system you can neither erase nor change any data, once recorded. Figure 1 clearly proves that traceability is an integral part of cognition, inseparable from it, and that in this design a correction is impossible, because the past cannot be changed. Similarly, a hard update operation, by rewriting data in place, would contradict the essential function of a memory system. Rewriting is related to macro expansion, a (deliberately forgetful) process known to be difficult both to master and theorise. Here instead, the process is unfolded, one may specify a view with arbitrary transformations to apply to any previous record. Given this mechanism, most operations become surprisingly simple, because everything written is immutable; thus, if something works today, it will work forever. (Suppose that necessary hardware architectures can be emulated or virtualised.) In the computer world, an immense amount of time, energy, infrastructure management, software sophistication is spent on the disastrous consequences of updating in place and its ensuing complexities. Consider how much relief came with the undo function in text editors. It takes the load off the users' brain to know they cannot do any harm to their data; it releases their stream of consciousness. There is no potential for 'inconsistent updates' when updating-in-place is avoided: layers and layers of comments may overlay one another, even contradicting each other, but without inconsistency. Time t is reversible.

Lack of trace is painful [15]: ''Those who cannot remember the past are condemned to repeat it'', just as happens in bugs by regression.

Plan 9 is a living demonstration of the soundness of suppressing update, at a certain level, in its main file system, allowing the *yesterday* command to look into the system's past,

usual*with*the In his ''Debugging backwards in time'', Bil Lewis [8] also shows the power of reversible t: by recording every state change in the run of a program, you can navigate the unfolding of every bit of information that might be useful (just as all events in α cover all the possible sources of information, for ever).

7. Letting the user unobtrusively express his own order and trace it at desired detail, and the generality of references

A single reference mechanism, at the level of records, accounts for citation, quotation, annotation, comment, or extracting data to feed into a process.

All the expressed order, all the structure is formed within records (especially as views), except for the unavoidable temporal sequentiality. It is the only place for variation, and it gives complete freedom: arbitrary structures can be expressed. Current systems tend to mix places where order is expressed, and maintain a confusion between stating ''In my present opinion, this piece of information is incorrect, because such and such'', and the urge to rewrite the past. This approach is rather different from much work on file systems with semantic concern [5], though not necessarily incompatible with it. Reference and annotation, not updating in place, is the general case, and appropriate to traceability. The essential thing is to let the user specify his own order, rather than impose an arbitrary one, and to ensure traceability in a non obtrusive way.

In cognition, selected events are recorded, and structured into one's own order. Some events are left dormant, some are continuously reactivated and reused, but few if any are totally forgotten. Cognitive psychology and perception theories consistently see the world as a chaos — in the sense that its structure is foreign — that each subject is challenged to organise, to render consistent for himself. (The chaos was faithfully represented in Figure 1 by an amorphous set in α.) When represented in an external memory, that organisation is also personal.

The very nature of a personal work is to impose one's own order on things. Generations of historians can follow one another working on the same corpus without repeating themselves; in this type of scholarship the essential contribution of each is offering his own model to interpret data, rather than simply applying a borrowed model. For example, many scholars reorganise their bookshelves for their current project, and workers often keep a selection of books or papers at hand, on a desk.

The decision of what to retain is a recurring one, and all too easy to overlook. For example, you receive a Sibylline email with a URL ''explaining it all''. The email was archived, but years later it no longer makes sense since your sender forgot to save the world simultaneously, and the URL vanished. If a reference is really useful, its name alone can do nothing: something must be retained of its content.

So far we have treated the architecture displayed in the diagram as entirely devoted to a personal repository. But since the structure with its triplet ID is absolutely minimal, it is not surprising that it extends smoothly to a collective use. There is no operational gap between personal and collective work in this architecture. The result of contemporary or delayed annotations is a fairly thorough record of the cognitive interaction of users. Users can use the same repository independently without harm, if they wish. There is in principle no obstacle to a very large number of users. Exporting the trace of a record is easy, by capturing all the cascade of dependencies (which can be very large if a view says, for instance, that it discarded very many records for reasons it extensively exposes as justifications). Records from different repositories can easily be mixed and disentangled. The *write once,* sequential structure makes it easier to do distributed replication. Backlinks are easy to find with an index, much as the web is indexed for search engines. The index can grow incrementally, and be recorded just as any other data.

Having all the structure expressed in records has an important consequence on the use of names. Names given at the birth of an object are important, for files as well as for variable

names in programs. But new names are called for when going into foreign languages (or staying in the same place, waiting long enough). References and views allow naming, assigning names appropriate to the current use for a file, a set of files or portion of a file, or a set of portions… any describable structure. Views open the possibility of a unifying namespace that could stand for centuries, evolving but always ultimately referring to the same original collection of data. Just as we have seen that metadata (relating two records) is by nature always an afterthought, appropriate names, which are just a particular type of metadata, are often found belatedly. A new way to *view* things is often a new way to name them. Certain names come to be accepted as labels, which restrains both the proliferation of names and the complexity of their relationships. A synthetic view can also cut back complexity.

8. More on personal vs. central repositories

There is more to justify personal repositories. ''Information empires'', where data stores are more and more concentrated, cannot be blindly relied on as external memories, however self-confident their security, or however open their systems. Any technical organisation can break. Natural hazards lurk. Empires by definition are subject to radical decisions. It is reported that Qin Shi Huang, the first emperor of China, had all existing books not complying with Qin historians burned (with exceptions for a few special fields), and later buried alive scholars who either still owned those books or still had them in mind and could spread their ideas. The library at Alexandria was destroyed. Information empires today are not in a different position. In mid April 2011, Google announced that on 29 April, 2011, files hosted on Google Video would no longer play, and completely shut down the service on 13 May [1], thus disproving those who conceived of video sharing as an archival solution. Users, if still alive, had that short time to hear about the shutdown, and quickly download their files again, because no archives were to be kept.

In the same vein, many initiatives of data preservation are focused on large data sets [7]. Among many reasons, large projects make it easier to attract funds, and to impose on their users constraining technical choices to ease administration. But the problem is actually to capture people's work the way they work: that is, how they interact with data. Any one seeking today for a long-term digital preservation solution on a personal basis is left without a solution, even without mentioning the inability really to address the concern for preservation in everyday activity. But preservation, archiving, cannot be an afterthought, all the more so in the digital world. The history of science teaches us that the most surprising breakthroughs come from the most unexpected conditions, and they are by nature impossible to predict. If we want to capture them, we need a solution to hand. Meanwhile, a huge amount of knowledge is continuously being built by hosts of individual researchers around many small, valuable data sets. Then some day scientists retire, and the traces of their knowledge retire with them. And it is not uncommon that a few month after submitting a paper, traces of how to reproduce the results are lost [2].

In the scientific field, we witness in practise that the more computers become intimate at every stage of a scientist's work, the less a given piece of work becomes reproducible, whereas we might expect the exact contrary. The creeping of instant messaging into scientific work, and the occasional vanishing of data on the Web, challenge more and more one's ability to capture them efficiently, and to retain appropriate images to make them usable as reliable references.

By reading a novel, one can experience someone else's slice of life, without having to actually live it. By borrowing someone else's library, one can watch through his cultural window. By borrowing someone else's repository, one can go even further, with the choice of jumping to conclusions, and backtracking freely to the sources, nearly walking by the stream of thoughts of a forerunner. The projected system, serving as a general digital laboratory notebook, would give that incredible ability to follow in someone's footsteps. If we want to ''climb on the shoulders of giants'', we must be able first to follow in their

footsteps (in their tracks). As the printing and dispersion of books have added demonstration to common sense, many copies make data safer [13]. But first we need solid, provable *personal* digital memory systems that are easy to manage.

9. Implementing a linear (sequential), write once file system

This section is even more a call for comments than the rest of this paper. It should now be obvious that an adaptation of Fossil [11] reaches the specification we aim at.

There will be, side by side, a user's workspace (home directory) together with the set of files necessary to boot the operating system into a particular, functional state, and a specific deposit area in which to put files or directories that will become a repository record by a sort of *on-trigger* prompted snapshot process that builds a specific Vac file with the attached triplet (*uid, host, t*). (A typical t would be an integer count from an arbitrary origin, convertible in calendar time; note that for instance 64 bit at a nanosecond resolution represents about 584 years.) Venti need not be changed.

It is up to the user to determine the chunks of data (e.g., a whole directory or many separate files) that are to become records; the main rationale is his own convenience. The semantics of 'gathering files' initially is not particularly strong and is likely to be split into more relevant sets in the future.

Eventually all the files are found in the repository. It seems that the notion of a home directory is necessary to hold at least a list of maintained views on what my work has been so far, or what my workplace now is. But in fact, a single view is enough, and perhaps experience will show that the distinction of a home is unnecessary and it suffices to fetch the last personal view from the repository. A view might also allow fetching all the bindings needed to boot a particular operating system state and file system configuration. As a result, only a transient workspace where current work takes place might be enough. Caching that workspace, to have it at hand immediately when one returns to the machine, is close to having a home directory.

Of course some bootstrap is required to first explore enough of the repository to be able to load and jump into another machine. Having the full hardware-software stack available is the definitive—the *best possible*—condition for allowing a user to fully interact with a given file. For preservation purposes, confining the main work inside a virtual machine allows saving and restoring that complete hardware-software stack. Most of the work should unfold on the repository, but determining how much so is reasonable, still needs to be determined by experiment.

The sequential record structure means that cutting a large repository into stripes as it grows to huge sizes is easy. Parallel access is then possible, which conversely results in duplicating some blocks, with an independent Venti for each stripe.

We assume it is or will be possible to monitor media memories for degradation (which is eased on a Venti-based storage system), and migrate them accordingly, thus ensuring continued access to the bits of a file. Well-traced, workable, computable corpora as planned here will undoubtedly be better candidates for the effort.

10. A language for the elementary interaction and its text editor

References—(hyper)links—are expressed as text, pointing to a record by a triplet uid, host, t, and a position. In a text file, a position is a simple integer (if you count in characters, different from bytes in UTF-8), but segments of text can be located by any other mean, like a string search, a regexp, or using the patterns and region intervals of ''lightweight structured text processing'' [9]. (Remember that if a string search leads unambiguously to a position, it will always, since files don't change.) In an image or other special formats, appropriate locators must be used.

As a minimal effort implementation we use Emacs' Org mode from a Linux system. Org is

a general purpose tool and a coordinated light syntax (in plain text files) for keeping notes, outlining text, internal and external links, including source code blocks and evaluating them, and converting them into source files using the noweb convention of literate programming [6]. Exports to various formats including LaTeX are easy, it also comes with many useful facilities as timestamping and calendrical calculations (so useful to historians), agenda views, tagging, maintaining to-do lists, very fit for project planning and perfect for keeping a log book, and at the moment, to give a feeling of what could be the work on a log book structure.

The aim at this stage is to do the work mainly from a Linux installation, with great use of Emacs (we know we are in a state of sin, but it compensates for many gaps of integration between the operating system and its windows). All the repository is kept on file systems under Plan 9. Nearly all the facilities we need are present under Emacs, or otherwise relatively easy to develop. Commands like find-dired or find-grep-dired in Emacs can give users a good intuition of what a view might be, in general, and of the power of text manipulation. There are, however, exciting prospects in a combination of features from Acme and Emacs, under the Plan 9 namespace.

Linux is not too exotic for a fully-fledged environment for a large set of potential users, and is becoming Plan 9 tamed. It is realistic not to expect people to shift from their habits, if we look more widely for future users, especially in the humanities or other fields ''foreign'' to computer science, to experiment with the system. One of the ultimate purposes is to be able to tempt users of any operating system with a facility for traceability that mildly changes their habits, though inevitably some systems will offer a result with rougher edges than others.

This combination we aim at offers the least-effort path to a minimal working installation of what is meant as a general purpose environment, as seamless as possible from programming to common use. It is difficult and useless to try to foresee what usage patterns will emerge, and nothing can replace hands-on experience. For instance, whether there should be some sort of reserved keyword to state ''This is a correction'' is left open, though intuition suggests that we ought to avoid it: proper computer semantics can reside in the repository's scripts, and fine human semantics can be written as subtly as poetry, in an accompanying comment. And natural languages evolve… Leaving the system widely open to multilingualism today, is to make it ready to face the future.

11. Concluding remarks

This paper is an attempt to define and justify the design of a digital system for traceability on a very long term. We have felt necessary to present an analysis in very abstract terms, axiomatic in spirit, and to connect each aspect with illustrations from a large range of cultural fields. It appears that embracing centuries consistently leads us to some fundamentals of cognition and cultural heritage, and perhaps paradoxically to the minute details of day-to-day work.

Early on, this diagnosis led one of us to seek a solution in operating system software research, and Plan 9 emerged as a prospect. After all, an operating system is a set of programs that make it easy for computer users and programmers to do their job [14]. We hope we have reached a sound, convincing outline, and identified some important essentials of working with a computer (and other sorts of memory devices!). Others should appear in the process of experimenting with a first implementation of the system. We hope that the design is clear, minimal and well-layered enough that we can cope with such surprises.

Our firm belief is that there is a favourable, exclusive niche position for Plan 9 on this topic, from where could emerge opportunities to spread like a virus. We submit that reflection to the Plan 9 community.

12. Acknowledgement

We thank Danny Lo Seen and Adrian Custer for reviewing early stages of this paper, Tristan Allouis for his commentaries on the figure. We are extremely grateful to Charles Forsyth who volunteered some editing, and ended up with a masterly rewrite. We found our words, translocated just in the right place. We are unfair not to count him as an author.

References

[1] Mark Brown. *Archivists step in as Google Video shuts down for good.* http://www.wired.co.uk/news/archive/2011-04/18/google-video-termination, 18 April 2011.

[2] J. B. Buckheit, D. L. Donoho. *WaveLab and Reproducible Research.* Dept. of Statistics, Stanford University, Tech. Rep. 474, 1995.

[3] DOI Factsheet. *Managing Data Relationships Using DOI® Resolution, Version 1.0.* http://www.doi.org/factsheets/ManagingDataRelationships.html, The International DOI foundation, accessed October 13, 2010.

[4] Michael Factor, Kalman Meth, Dalit Naor, Ohad Rodeh, Julian Satran. *Object Storage: The Future Building Block for Storage Systems, a position paper.* In: proceedings of the Second International IEEE Symposium on Emergence of Globally Distributed Data, June 19-24, 2005, Sardinia, Italy.

[5] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, James W. O'Toole, Jr. *Semantic file systems.* In: Proceedings of the 3th ACM symposium on Operating systems principles, October, 1991.

[6] Donald E. Knuth. *Literate programming.* CSLI Lecture notes, 27, Center for the study of language and information, Stanford, California, USA, 1992.

[7] Kyong-Ho Lee, Oliver Slattery, Richang Lu, Xiao Tang, Victor McCrary. *The state of the art and practice in digital preservation.* Journal of research of the National institute of standards and technology, 107, 1, 2002, pp. 93--106.

[8] Bil Lewis. *Debugging backwards in time.* In: Proceedings of the Fifth international workshop on automated and algorithmic debugging (AADEBUG), Ghent, Belgium, September, 2003, Michiel Ronssea, Koen De Bosschere (Eds.), pp. 225-235. http://arxiv.org/pdf/cs.SE/0310016, [2010-10-14 Thu. 10:28].

[9] Robert C. Miller, Brad A. Myers. *Lightweight Structured Text Processing.* In: Proceedings of the 1999 USENIX Annual Technical Conference, June 6-11, 1999, Monterey, California, USA, pp. 131-144.

[10] Sean Quinlan, Sean Dorward. *Venti: A new approach to archival data storage.* In FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies, page 7, Berkeley, CA, USA, 2002. USENIX Association.

[11] Sean Quinlan, Jim McKie, Russ Cox. *Fossil, an archival file server.* Plan 9 user's manual, volume 2, fourth edition, 2002.

[12] Sean Rhea, Russ Cox, Alex Pesterev. *Fast, inexpensive content-addressed storage in Foundation.* In: Proceedings of the USENIX Annual Technical Conference 2008, June 22-27, Boston, Massachusetts, USA, pp. 143-156

[13] David S.H. Rosenthal. *LOCKSS: Lots Of Copies Keep Stuff Safe.* Presented to the NIST Digital Preservation Interoperability Framework Workshop, March 29-31, 2010.

[14] Jerome H. Saltzer, M. Frans Kaashoek. *Principles of Computer System Design: An Introduction.* Morgan Kaufman, Burlington, MA, USA, 2009.

[15] George Santayana. *Reason in common sense.* Dover Publications, Inc. New York, 1980 (first published by Charles Scribner's Sons, 1905).

[16] Denise Schmandt-Besserat. *Before writing, volume 1, from counting to cuneiform.* University of Texas Press, Austin, Texas, USA, 1992.

[17] Ben-Michael Schueler. *Update reconsidered.* In: Architecture and models in data base management: proceedings of the IFIP conference on modelling in data base management systems (Nice, France, 3--7 January 1977). G.~M. Nijssen, (Ed.), North Holland, Amsterdam, 1977, pp. 149-164.

[18] T. R. Schellenberg. *The management of archives.* Columbia University Press, 1965.