KeyWorx: A Working-Alone-Together Reflection

Collaborative Practice on a Collaborative Product

Introduction

An historical approach to the methodology driving the development of artist software by a largely non-profit European cultural institution would entail a chronicling of hundreds of planning meetings, highly charged, emotional debates over lo and hi end software design, tens of thousands of working hours, financing schemes, open source arguments, event production, intern training, workshops, demo disasters, artist support, survival strategies, frustration, euphoria – the gamut.

The KeyWorx development project has had, since 1996, a slowly escalating trajectory supported with pioneering conviction by the Waag Society for Old and New Media in Amsterdam\(^1\) with early help from cultural funding agencies.\(^2\)

I have opted to condense a reflection on this five-year process to remarks from the three main programmers, Niels Bogaards, Just van den Broecke and Tom Demeyer, as their perspective is often overshadowed in discussions of artist software and technologies by the aesthetic aims of the development. In fact, they are inseparable and co-dependent. I have also included remarks by Marleen Stikker, Director of the Waag Society, for her insight on sustainability of art/tech research projects.

---

\(^1\) [http://www.waag.org](http://www.waag.org)

\(^2\) KeyWorx has received support from Stichting STEIM, de Amsterams Fonds voor de Kunst, de Mondriaan Stichting, Fonds voor de Podium Kunst, The ArtsAlliance, UK and MultiMediaLab2, UK, VSB Fonds, Prins Bernard Fonds and Stichting Doen
I hope to convolve three R&D areas, coding practice, artistic/design practice and management by alluding to the seminal McLuhanist ‘the medium is the message’ (the process is the product, you are what you eat) angle. This approach begs several questions within the context of artist software development and these were put to the programmers of the technology for their point-of-view.

- How is the development methodology mirrored in the product?
- Specifically, in the case of KeyWorx, how and where is the collaborative development process, the making of together, evident in the design, functionality and aesthetic of the application which is, by definition, a collaborative tool?
- Are the strengths, weaknesses, idiosyncrasies, styles and philosophies of the development team present in the functionality of the tool? How it interfaces with the user?
- Is the design practice of modularity in software architecture analogous to modular working methods analogous to a modular user environment?
- How does the application theoretically and practically enable 'collaboration'?

Collaborative Process

The *a priori* premise that artists would want to create performances together in a shared, egalitarian environment was both timely and naïve. The new politics of authorship and open source merging with the new science of complex system theory, emergence and connected cognition is seemingly fertile ground for distributed multi-user projects that seek to ratchet the genre up from online shoot-em-ups to interactive, multi-modal creative expression. What looms ahead as the enabling technologies mature is the all too human capacity to communicate, negotiate, abdicate, exonerate, etc., etc., with the additional handicap/advantage of physical distance.

From my perspective, these issues raise a question on the very nature of the recursive collaborative process; specifically, how the KeyWorx team ‘collaborated’ on a blue sky technology designed to enable ‘collaboration’ between interdisciplinary artists and how that gestation is interfaced to the end-users. To what extent does the communication between the development team and the architectural model determine how the end-users will access and recontextualize this experience? That the team unanimously feels the development process would have been enhanced by working in physical proximity is some evidence of the imminent learning curve in quality, remote (synchronous) interaction. Perhaps it is an indication of proximity curves in higher level dynamical systems? It is also
evidence of the infrequency of our collective use of the application we’re making which
would imply that we are not (all) making it for ourselves, that between us we evaluate our
work from objectively critical and subjectively passionate microscopes.

A Brief History

In 1996, a group of four artists approached the fledgling Society for Old and New Media to
sponsor a proposal to create a tool enabling interdisciplinary artists to collaborate in a
virtual studio space on the Internet. Supported by the Amsterdamse Fonds voor de Kunst
the project began to take shape and refine it’s goals in 1997, under the guidance of
Marleen Stikker and Carolien Nevejan, then co-directors of the SONM.

As the initiator of the project, I was anxious to recruit Tom Demeyer, then from Stichting
STEIM³ in Amsterdam as a developer. His combined work with real time image analysis
(BigEye and Image/ine) was at that time, unparalleled in artist softwares. The vision for the
research had expanded from a realizable (by ’96 standards) multi-user KeyWorx controller
of bitmapped graphical images to a distributed multi modal synchronous media
synthesizer, anticipating interactive video and audio streaming. A visionary project in the
heady days of the late ‘90’s. Marleen Stikker was able to coax Just van den Broecke, fresh
from multi user research at Lucent Technologies, to join the project. With the additional
good fortune of finding a remarkable young intern, Niels Bogaards, as the programmer of
the user interface; the project was launched in January ’98. My role was to design the
preliminary user interface, define and present the concept to the public, play the initial
artist developer role and draft the subsidy proposals.

The language of an early subsidy request description attempted to find an appropriate
analogy for real-time filtering processes that were largely unknown to the arts practice
funding committees at that time.

At its inception in 1996, the creative team of this project envisioned a tool that
would allow online multi-user collaboration in a ‘synaesthetic' environment - sound
influencing visuals, visuals influencing sound. Interest in synesthesia - morphing
sensory input (vision, sound, taste, touch, smell) is centuries old but only a handful
of people, clinically labeled 'synaesthetes' (originally thought to be ten in one
million but now thought to be one in two thousand), have the ability to truly

³ http://www.steim.nl
experience the taste of a sound, for example, or the shape of smell. For the rest of us, new media technologies are creating possibilities to experience the interaction and direct manipulation of combined sensory activity. Modern biosensors allow muscle movement, brain activity, heart rate, etc., to control audio sources just as these same audio sources can, for instance, manipulate video imagery. ...The current and rapidly expanding capabilities of media technology are making these types of cross-synthesis possible and accessible to a coterie of scientists and media artists and to the general public. That digital tools and all those ubiquitous one’s and zero’s have opened new areas of sensory exploration is both ironic and expected; recalling Pythagorean visions of a world described by numbers and symmetries that transparently resonate in the nature around us. Add to this mix the potentiality of real time group collaboration facilitated by linked environments and shared 'canvases' and we embark on a fascinating new frontier."

What is it?

KeyWorx combines a distributed multi-user, multi channel environment with dynamic cross-media synthesis, providing the tools for extensible forms of telecommunication, telematics, interactive broadband and collaborative performance. KeyWorxs' ability to synchronously synthesize media with multiple users in a common workspace makes it a powerful live performance tool for remote interaction and interdisciplinary work.

KeyWorx is a next generation media synthesizer, meaning that any digitizable input or medium can be broken down to its component properties and those parameters can control or effect all other properties of media or dedicated electronic devices.

...Creativity is an integral and evolutionary catalyst of this dynamic process - a kind of randomizing engine that produces change, growth and unpredictability. In KeyWorx the users provide this element by choosing and filtering connections in a shared, realtime environment that can be both scrupulously determined and unforeseen. The users/players themselves form a nested collective as actions made by one effect the experience of the whole (performance) in obvious and/or subtle ways over the duration of play. A small alteration by one player in the value of one component media property can create significant ramifications in the performance,
exhibiting a ‘sensitivity to initial conditions’ otherwise known as the butterfly effect of chaos theory.

The unpredictability of the synthetic process is certainly a provocative element in the play of KeyWorx. But the truly maverick agent in the process is the group dynamic. Will disparate communities form around issues of friendly improvisation or structured linear development? Will artists routinely compete for control or abdicate control in favor of a defined group aesthetic? Will we face an extensive learning curve in collaborative methods and etiquette? To what extent will users of shared environments (beyond the current reach of chat sessions) tolerate the invasiveness of other users? How will we respond to sensory cacaphony and will the nurturing of synaesthetic capability engender new forms of art and communication? How will we develop our sensitivity in 'listening' to each other within a shared workspace and what are the semiotics of non-linguistic exchange?iii
WAT Methodology

The Working Alone Together (WAT) practice, modular methods for a modular architecture, adopted by this team for pragmatic, financial and personal (established working patterns) reasons in the early years of the project represents a curious corollary to the WAT aspirations of distributed collaborative learning and performance software environments. There has been much research and effort in past five years focused on issues embedded in emerging telecommunications practice in education, in the arts, in the workplace. A number of projects are represented in this book. Levels of perception and cognition in telepresent/telematic environments, degrees of engagement and interactive agency, are all factors in this discourse.

The questions raised by reflecting on the relative successes and failures of a working process are perhaps of more interest than the modestly documented conclusions.

- Is remote collaboration effective?
- Can the confluence of discreet streams of activity result in a sustainable economy, a well-oiled engine, a cohesive collective effort?
- Is the routine practice of connected collaboration something we would ever opt for over proximate collaboration?
- Under what conditions do we work and create best together?
- What have we learned?

That when polled, the KS team unanimously felt it would have expedited the project if they were all working in the same room is a poignant remark on the expectations of the project itself which include the facilitating of remote collaboration. Ostensibly, as mentioned in the interviews, the programming process is dissimilar to the user experience of the software, so a corollary is misguided. Nonetheless, as practice within practice it's of interest and alludes to terminology offered by van den Broecke – ‘minimum coupling – maximum cohesion’.
Diagram of KeyWorx’s Architecture

The Programmers POV

*I interviewed the three core programmers individually and asked questions about the development process that correlate with the desired functionality of the application. Many of the bullet point questions mentioned earlier are addressed.*
**Tom Demeyer - The Realizer Component**

SD - What is the function of your component – the Realizer, in framework?
TD - Everything that’s real time. Dealing with real time video and audio generation and effects, sensor inputs, internal communication. Basically the module interface, module API, the low end internal structure of the realizer.

SD - How happy are you with choices made in early’98 given the parallel development of protocols, bandwidth, standards, etc. What would you do differently with hindsight?
TD - I don't think we made any wrong decisions. I think everything we decided then was quite ok.....

SD - What was particularly smart of the team?
TD - Just's XML idea was very good. It was just then coming up and wasn't as hyped as it is now. In retrospect it was a very smart idea. Also the team consciously implementing it throughout the whole architecture, not just as communication but also as a file protocol, consistently using it in every nook and cranny of the software. The modular approach is an obvious one.... it wouldn't be as flexible as it is now if we hadn't approached it in a modular way.

SD - Are there any negative aspects to the ‘modularity’ of the process – three programmers working on three modular components?
TD - That 'modularity' doesn't have any negative aspects. Neither of the two modularity’s do. Even if you approach it from the perspective of open source - what advantages and disadvantages do you have? You don't have any disadvantages to structuring your code well. It's always advantageous.

SD - How is the process mirroring what we now have in KeyWorx?
TD - The process is mirroring the characters of the people, mirroring the code. We're solitary programmers (well, Just less so). We've always programmed by ourselves, we've never been part of a big team of programmers - this is just a clever trick of avoiding that. We just do our own thing while we're still part of a team.

SD - How do you perceive the process of development reflected in KeyWorx?
TD - I think it has ramifications. If we really worked together the whole time we wouldn't have as many problems as we do in the multi user department because we’d be testing much more while we're doing it. Since we're working alone, you may have this good idea and work a long time but when it all has to come together it fails and it turns out that wasn’t ever possible.
I think this might be the central disadvantage to working this way ....in working alone the immediate testing phase is postponed and that's a cause for trouble and a cause for inefficiency - that's probably why it progresses so slowly, this multi user debugging - because all the programmers are needed to do this, it's virtually impossible to get everybody working on the same thing at the same time.

SD - So if you're working on a multi user application you should be developing in a multi user environment yourselves.

TD - Yes. Or you have to set a very rigid multi computer environment for yourselves. When you write code you're a very casual user as well, on submicroscopic bits of the code, but you're using it to test the validity of the code, the assumptions, but you're using it microscopically, not anything like an end user would use it. It's just small tests. In a multi-user bit of code it's not possible to do it yourself.

In these kind of programming environments like de Waag it's always true that the process reflects also the character of the people, the way they work together - it all reflects on the end product, there's no question about that. That's why you won't ever see big corporate structures writing innovative software.

SD - Open source? In principle? For KeyWorx?

TD - I'm all for if it's going to be effective. It has to be effective for us, that's the criterion. Our angle is if we make it open source we hope that it will improve. For that there are a number of preconditions. To meet those conditions we have a lot of work ahead of us. If you stick this code as it is now on the web as a public source repository, you'll have lots of people picking bits and pieces out of it for their own projects, which is fine, but you won't get anything in return because there won't be anybody who oversees the whole thing.

You have to oversee the whole thing in order to concentrate on a detail. In order to effectively concentrate you have to know how it fits into the whole thing.

SD - Does the modularity help or hinder that?

TD - The modularity makes it less important to oversee the whole thing, less critical to know the big structure of the whole thing.

But we should address it. If we all agree that it would be nice if we have a bigger input from the rest of the world then we should address it now.... but we first need to make a kind of nervous system for all the modules to attach to.
SD - In ’97 you were very negative about Internet development projects. I remember you saying when I first approached you about this project – ‘oh the Internet – why would you want to get involved in that?’ Have you changed your mind?
TD - I haven’t changed my mind in that I still think it’s impossible to do proper art across the internet. Old fashioned art – music or video, I still think that’s far-fetched or not possible and not very interesting. New modalities- sure, I’m all for it but they have to be new ways and that’s very hard because we still don’t see them. I still haven’t seen it. Not something beautiful, new, which couldn't have been done without the Internet....What I have seen is old media in new frameworks and usually suffering in the process. It’s just been the decline of art but I’m still hoping.... I have not yet seen anything traditional across the Internet or being used collaboratively and not suffering in the process. I enjoy listening to a string quartet more than a cod I enjoy a cod more than an online jam session. It’s a matter of quality. Not necessarily quality of content but of technical quality, which is important to me.

SD - What your interested in interaction with artists is not just enabling them but being challenged in return?
TD - Absolutely. Otherwise I wouldn’t do it.
SD - You don't like to work on something unless it's a challenge.
TD - It should be something just out of my reach.

Just van den Broecke – The Server Component

SD - What exactly is your function in the framework? Your responsible for the Server component, what does that entail?
JvdB - First one step back. We're all four responsible for both the conceptual and technical architecture. One of the strengths was that we actually sat down to work out the technical architecture. And that is very important. ... Broadly we have three processes – we have the Patcher, the Realizer and the Server. Some of these even have an internal architecture which is component based in the sense that you can add plug ins – that makes it flexible. Of course you inherit this and it's very hard to change that architecture. It's also the strength. We have a reasonable architecture.

SD - In hindsight, would you change any of the major early decisions?
JvdB - No. I think everything is perfect.
There are a few one-liners about software architecture. *Conceptual integrity.* Organizational architecture should reflect software architecture Tom is working on the realizer and I on the server. What it means is that we have the least possible coupling. And here is the next one liner - In software architecture coupling and cohesion are important in the sense that you must minimize coupling and maximize cohesion. So typically a component should be cohesive in the sense that it does a set of well-defined functions. And that it should have as couplings with other components. By each working on a separate process with well-defined protocols, you have at least a well-defined coupling because if you work together in the same piece of code, then you have much more coupling.

SD – But there has been some duplication of effort?
JvdB – If I think back I think we could have made much more reuse of effort. I think the architecture between the realizer and the patcher has lots of room for improvement. On the other hand, as soon as you have reuse – a piece of common code – you create more dependencies on that code, which in some cases can be fragile in the sense that when you change the piece of common code the users will get failures. While if you have separated everything at the expense of some duplication than you don't influence each other. It's always a trade off. For pieces like xml parsing, protocol client libraries, they could be very much improved.

SD – How is it that you three programmers have collaborated on a piece of collaborative software? How much is that process reflected KeyWorx? In this particular case you have a collaborating group of people working on an application that's about collaboration.
JvdB – for one thing, the multi user aspect has been greatly underestimated and for some people, something like a side artifact. ...For example, the Realizer, to put it bluntly, it's another interrupt... I had the same thing when I started working on multi user because I came into this group at AT&T and moved internally to this other group (working on multi user) and said 'What's the big deal? You send some stuff to each other. Why are you working with hundreds of people on this with these thick documents?' But when I started working on it I realized, for one thing, what I underestimated was the number of failure modes that can arise.... At that time we developed user interfaces as a side effect to access the system, not starting with users first. Maybe that's what also happened here. There were already so many issues to solve. And for me, from a network point of view, I tend to develop from the inside out. But to answer your question, I think the approach is reflected in the product.
...with this particular application, I wonder if there shouldn’t have been more play, or interplay from the very beginning amongst the team?

JvdB - It’s also something that happens a lot with with programmers – they make something for someone else but again this is also a criterion for good architecture – eat your own dog food – would you use this yourself as a product. I think our intention is to do so.

SD - there's a dynamic that would happen if it were the four of us or even just the three programmers, there would be things you’d see immediately in the software that...

JvdB - but as a software programmer you approach an application differently than a performer.

SD - Do you think it would have made a difference if everyone were sitting in the same room?

JvdB - Communication is one of the most important aspects in software development and of course lack of communication is the reasons most projects fail, apart from exaggerated requirements and schedules, but we didn't really have a schedule. I think it's better to have a shorter stretch of time and work five days a week in the same room.

SD - How would you rate the experience of working on an ambitious piece of software between a corporate client and a non-profit cultural institution?

JvdB - There is a difference working formally than informally that could coincide with working commercially or non-commercially. It's not the case that all commercial organizations work formally. Put bluntly, I see everywhere the same mess. Even if they try to work formally. Again one of the critical issues is communication and architecture design.

The difference is more freedom in making changes, when you like to have things changed, without going through a formal process. Every software development is about iteration, when you make decisions at the beginning, you say oh, it's wrong, you go back, ...even in the commercial business they are also trying to formalize iteration processes...

SD - Do you have a preference?

JvdB - It's more fun working in this environment. At the same time, what’s needed is a more agile or lightweight methodology. A formal process, I've hardly seen that working. Anything you develop new, and to try to put an existing process around it, it just doesn't work. One size doesn't fit all. You need to develop a process particular to your
organization, particular to your project. Maybe we could have spent more time thinking of the process.

SD - Open source. What are your feelings in general? With KeyWorx?
JvdB - It’s a blessing. We al use it without even knowing, 80% of services use apache. It works the best if you make some kind of extensible framework product with a well-defined architecture and if you somehow have a way of earning money from services. That’s the model. That’s the case of course with Apache and Linux. They both have an established modular architecture. For one thing, because you have so many eyeballs looking at that software, the quality is improved. If it’s a product at the right place and time, people will participate, not only in writing but also in writing manuals.

SD - Can you open source the three modular components?
JvdB - You need a lot of ironing out of the architecture, making everything understandable and documented. You need to manage it as a project. You need version management in place. It would attract more users. The fear is that someone takes the product and makes it a commercial product. But if the momentum is behind the open source then…
It’s a way of giving back to the community. Otherwise you’re only consuming.

**Niels Bogaards - The Patcher Component**

NB - I work on the Patcher. It’s the way to build your multimedia patches, your instrument. I have very strong opinions on how you should make multimedia instruments. It’s thinking about that and then implementing. Implementing an interface is very difficult and prone to errors as it’s the part where you’re most directly interacting with the user and you don’t really know what the user is going to do so you can’t really test it. That makes it exciting.

SD - Your function on the team extends further than the Patcher component?
NB - As it happens I’m the one on the team that has the most time. If you want stability and sophistication, then someone has to spend time verifying the integration of the components of the program. Otherwise you can have a really nice interface and a buggy engine. I’m interested in all parts of the software.

SD - How has the process effected, after years of development, what we have now?
NB - I think we have a team of very talented people with very bright ideas, that's why what we're making is at least in potential so exciting; it could be such an important program and breakthrough. The fact that we spent almost a year thinking about the application when we started, I think that's still the best thing that we've done. When we really began implementing we knew what we wanted. That was a good thing. Unfortunately, we didn't right away set our technical boundaries so when we started we were thinking about making it cross platform with many interfaces. I think we wasted a lot of time and still do on these issues. For example, all three engines have their own parser for the xml data we exchange and of course we could have used just one.

In hindsight I think that it was really stupid that we didn't ahead of time say we know what we want to make and we're going to make real decisions on who it's for and what operating system it should run on and a bit of a schedule. And that's disappointing.

SD - would it have been better to be working in the same space?
NB - Sure. ...if you're all in the same room, or at least at the same level of intensity then you can also excite each other. Where now everyone has their island and you don't know what's happening on the other island more or less and then you're unsure if things that happen in the other parts are sound even. It leads to a situation where you think maybe it's better if I do that.

SD - How does the process effect a KeyWorx session? Does your frustration reflect in the design of the application?
NB - I don't have the same feeling because it's not the same people I'm in the KS sessions with and very often they're at the same or higher level of intensity than I am so that doesn't really apply. I still very much agree with the ideas behind the application. I think almost everything we decided after one year of thinking is still true and if it would all be there it's still relevant. I'm not at all disappointed with the genre or working with the application. It's the most exciting thing I know. ...I know that I can have real fun in KeyWorx and I know that others can.

We've been slow because we've been duplicating work and we have too little work intensity and too little hours of work, but also when we work it wasn't like everyone was really anxious to get it out. I know from working with other people in a band that if you really want it you just don't stop until you have it. For myself I've had that feeling that this is so important that we have to keep going. ...Something that's so novel and so on the edge, well, I can't even explain to myself how that’s possible– how can you say, let's
wait a week. My only other real multi user experience is being in bands. I think the only relevant variable is that everyone is as much involved as the others.

SD - You’re equating programming and writing code with a creating experience like playing music in a band with the adrenaline pumping?
NB - I think it’s like that for all of us. I think we all want to get things working. The excitement of the act itself is the same, but it’s too individual. It’s not like WE have made this, everyone is like, I made a part of this. This is my part and that’s a nice part. It’s not like this is our thing and I don’t care who made it as long as it works.

SD - Modularity. It makes sense in terms of the architecture but does it make sense in terms of the working method?
NB - It was really simple that we decided to go for this modularity by person, because everyone had to work in his extra hours, Saturday nights you’re adding features on and you have to do that at home, and you have to have your own domain you’re in charge of. And it’s experts that in their little niche field are so good that it’s a waste of time even to discuss it. But it all has to integrate.

SD - Open source. What are the advantages?
NB - I think the only reason to have it Not open source is that you may have things in there that you don’t want to share, either for reasons of making money (selling the source code) or that you’re ashamed of. You need to make sure that nastiest parts, which are probably the buggiest parts, are cleaned up. That’s a big advantage. Another big advantage is that once it’s open source it cannot die anymore. Anyone with enough will to do it can work on it and make sure it lives on. After we’ve worked on it for these years - it shouldn’t die, like we’ve seen many, many applications in the media world.

SD - Since KeyWorx is at this moment free, what advantage would open sourcing it have?
NB - Even though the software is free of charge, its use, at least up til now, does require a lot of effort from the users; we call all of our users testers to make them aware of the fact that they need to put in work and communicate with us. For artists, focused on realizing their artistic ideas, this can be a frustrating process. Being open source may send out a signal to users that they own the software as well, and may make them more willing to accept the fact that the level of stability is incomparable to some commercial products like Photoshop.
It’s all about collaboration, and the less money, rights etc. are involved, the happier I would be.
SD - Last year you said you were against going open source because you didn't feel your commented code was up to open source distribution standards.

NB - That's true but since then I've looked into it, I don't know if it's even that bad. I've seen a lot worse. We've made discoveries. I know that there's parts in there that may be functionally not so relevant but took me a week to discover how to do it. But ok, if this is open source then the next person won't have to take a week finding that….I know that I'm never going to sell that information and by the time I find a buyer for that information it's already obsolete.

Even tough the software is free of charge, it's use, at least up til now, does require a lot of effort from the users; we call all of or users testers to make them aware of the fact that they need to put in work and communicate with us. For artists, focussed on realizing their artistic ideas, this can be a frustrating process. Being open source may send out a signal to users that they own the software as well, and may make them more willing to accept the fact that the level of stability is incomparable to some commercial products like Photoshop. It's all about collaboration, and the less money, rights etc. are involved, the happier I would be.

SD - Do you prefer enabling artists or blue-sky research? Where is your sense of satisfaction?

NB - You can have theoretical bottlenecks and solve them and afterwards hear from users that - there's this one button and it has to be bigger otherwise I don't like the software. You cannot realize that looking at diagrams or lines of code. You have to do it. You have to be in a nervous situation where you need to have it working, working fast, working exactly the way you want it to and only then can you really understand what it's about I think.

*Marleen Stikker, Director Waag Society*

*Role in the project*

In the beginning my interest in KeyWorx was in the gaming aspects of multi-user environments and scenarios. Initially I was more involved in the concept of it but now I'm more involved in finding the support structure for software development within the Waag, and for KeyWorx in particular. Over the years it has become the technology
platform for other projects and applications in the Waag, especially for the educational sector.

Support structures

I consider software to be a cultural artifact, but there is no support structure for software development within the arts and cultural sector. The funding bodies finance events and installations but not the underlying sustainable structures for software development. They don’t have the technical competence to review a development project like KeyWorx. The more technology driven funding bodies like the IST programme of the EU do not recognize the R&D potential of Cultural labs like the Waag.

Open Source

Open source can be a very interesting option for the cultural sector. Unfortunatly, there is not yet a solution for a sustained financing model of cultural R&D. You can finance development by selling your services, but that is something different from being a R&D lab.

An issue that is mostly overlooked with open source is the need for a extremely hierarchical organisation. The charms and the leadership of the core developers are key to the success or failure. The worst thing that can happen to an open source project is that is ‘forks’ and you end up with competing software projects instead of accumulating knowledge.

I think it’s legitimate that the EU supports cultural and artistic software development. RADICAL is about that – trying to raise the awareness of the importance of cultural software development and the importance that it is has for innovation outside the arts world.

An End-Users POV
To conclude I'd like to pull back from the discussion and commentary on the internal creative, collaborative process and switch to the external process, or the way the application is perceived and utilized by the artists it is designed for. I will take a most recent example of a small telepresence/telematic research piece, Liveform:Telekinetics, by Michelle Teran and Jeff Mann of Toronto Canada.

On April 6th, 2002 a joint dinner party was held between Toronto and Amsterdam with approximately 14 guests at each location plus three special guest robotic devices. In her project proposal Michelle explains:

“LiveForm Telekinetics combines these areas in a research effort to define new techniques and technologies for telematic environments which bridge the virtual and the physical.
In the instance of the dinner party, KeyWorx was used as the backbone to experience the AV streams, send MIDI commands to the robotic devices, chat, mediate the images, merge the place settings, pour the wine, etc. Chefs at each location prepared the food without additional aid.

In a post-dinner ‘critique’ Michelle writes:

First of all it was a complete success on our side. As you say, everything worked, on so many levels.

What I am most pleased about was the absolute integration between the screen, the kinetic props, the physical space, live performers and food.

….It took us about 2 hours until we were in a “zone” where we could actually be communicating well. After a part (but not every single part) of the system was up and running, I could successfully play my role as a moderator b/w the two spaces, able to talk back and forth with Niels, and relay this information to the others at the table.

All the props were really effective:
- Rob’s bottle pourer was great
- The glass clinker immediately caused people to stop and wait for the speech
- The fish guests took more and more seriously until we were having intimate conversations with it (we’d speak into the mike, but be looking right at the fish, and then would wait patiently for the fish to respond)
- Tamara’s chandelier was extremely beautiful (too bad you couldn't hear it). We attached a midi sound module to it so that the chandelier moved and could also play notes at the same time. We finished off the afternoon gazing at the chandelier consisting of 32 Chinese lanterns that would spin and emit these wonderful musical tones.

The screen on the table, mixing the two QuickTime streams + text was extremely gorgeous, lovely, image, wonderful mixing space.

So, just a few comments to say that the experience was extremely wonderful and that I am personally very happy that we did this. I think we did manage to accomplish quite a few things and I’m very happy to be a part of it.
I'm also totally thrilled that we were able to do this project with the Waag and hope to have more Toronto/Amsterdam exchanges in the future.

Conclusion

That collaborative efforts are by nature ungeneralizable and subject to the chemistry of personality, the whims of mood, the weather and the price of milk is pretty much agreed. Sustained collaboration builds upon a wary foundation and formulates degrees of trust, weighted probabilities, and shared outcomes. The interpersonal dynamic, deep convictions and general anarchistic attitude of the KS team⁴ are embedded in the behaviors, style, function/dysfunction, language and philosphy of the technology. For example, there was

---

⁴ Extended team includes Lodewijk Loos, Eric Redlinger, Ben Soreé, Floor van Spaendonck, Klaas Hernamdt, Fokke de Jong, Michelle Teran and Robert Steijn
an early, conscious decision to NOT implement a moderator or director function in the application, requiring players in each KeyWorx session to negotiate or abandon hierarchical structures. In retrospect, this decision leaves open the possibility of emergent behaviour between the players (nodes) and provides a test model for adaptive collaborative dynamics.

There is more than code and design here. There is a mini-culture, at once elastic, cryptic, resilient, stubborn, inflated, modest, youthfull, aging, impatient, impulsive, cautious, bored and enervated. It is to the credit and vision of the Waag Society to sustain an ambitious experiment long enough for it catch its breath.

Marleen Stikker:
“This is the only project where people are really angry about certain decisions about software. I really love it. That makes so much sense. To me it’s what it are all about, that choices in software are not objective choices that they’re subjective and exclude certain possibilities and enable other possibilities. …That’s my interest and why I’m doing this. To me the KeyWorx team is the living example that software is culture.”

1 David Garcia, Barbara Pyle, Nanette Hoogslag and Sher Doruff
2 From a request to the Mondriaan Foundation February, 1998
3 Performance Research, Vol.4, No. 2 Summer 1999