12th February 2008

Fundamentally understanding IT? - Why Web 2.0 needs architects. Part II

This is my second blog entry related to my claim that Web 2.0 needs architects. It's been a while since the first one because I've been working on a new book on enterprise architecture (as also mentioned in the previous entry). This blog entry is to some extend a raw dump of some thoughts, and hasn't really been polished. I just didn't want to wait much longer. Nevertheless, I hope that it, even in its current form, sparks some discussions.

In my opinion, Web 2.0 is yet another technology trend in the list from OO, via CBD, ORB's, to The Web, SoA, XML, Semantic Web, Web 2.0, ... Thinking about the promises of Web 2.0, and the role of (enterprise) architects in this made me go back to an old question: ***What if we can create a paradigmatic model of IT in terms of which we can explain current IT trends and predict future trends, or even derive directions for future technology developments?*** Rather than having to be bystanders while one new technology passes by after the other, we would be able to al least position and predict some of these technologies, and possibly take the lead in creating new technologies. Mind you, one does indeed need to remain aware of any paradigm breakers.

As a conceptual modeller (ER, ORM, DEMO, SBVR, ...) I've always been interested in creating/finding an abstract model of information technology that allows us to generically reason about the abilities of technology irrespective of trends/technologies that pass. Around 2001, we were toying with the concept of an ActorWeb (see http://www.cwi.nl/events/conferences/Components/Proper.pdf [http://www.cwi.nl/events/conferences/Components/Proper.pdf]). This "we" consisted of some people from Ordina, Escador and CWI (Centrum voor Wiskunde en Informatica), while also having some interactions with Hans Goedvolk from Capgemini, who had coined the notion of Actor-based Information Systems. Regretfully, we have never been able to turn this into a proper research project, and since one of our other ideas was evolving into what is now known as the ArchiMate [http://www.archimate.org] project, we put this idea of an ActorWeb on ice.

The basic idea of an ActorWeb is to view IT around us as a set of actors interacting amongst each other as well as with the social and physical world. To understand what this might mean at a more concrete level, I've spent some time thinking up the following example model. Mind you, it's just intended as a suggestion to show how such a paradigm might allow us to position and analyse technologies.

Let's think of IT systems as a digital world harbouring a society of digital actors supported by a digital ecology. The actors interact to each other, they may have a memory and cognitive/computing abilities, and some may even interact with actors in the physical world. The society of digital actors lives in a digital ecology created in terms of hardware and low-level software (operating systems). The digital ecology does not always have to be a connected whole. So interactions between digital actors living in different parts of the ecology might sometimes be hampered.

Each actor in our "digital society" has, analogous to human society, certain abilities and properties. Let's presume all actors are able to communicate to other objects. Let's furthermore presume that the least thing they can communicate is their (unique) name. More advanced abilities would be:

1. the ability to cling to an object in the physical world (think RFID!)
2. the ability to cling to an object in the digital world.
3. the ability to take measures in the physical world (location, temperature, etcetera)
4. the ability to measure the proximity in the digital ecology to other actors in the digital world (e.g. two digital actors running on the same network-node versus two running on different nodes). The proximity might refer to

computational locations/nodes, but also to time.
5. the ability to measure the proximity in the physical world to other actors from the digital world (e.g. a digital actor in an RFID chip clung to a book passing by a sensor harbouring a digital actor being able to sense the physical proximity of digital actors living on RFID chips). This proximity might refer to physical location as well as time.
6. the ability to interact with actors in the physical world.
7. the ability to manipulate objects in the physical world (e.g. robots).
8. the ability to manipulate objects in the digital society (or even the ecology).
9. the ability to move around the digital ecology in such a way that the actor has a different physical location (i.e. moving from one server to another).
10. the ability to store large amounts of information.
11. the ability to reactively respond to stimuli.
12. the ability to pro-actively exhibit behaviour.
13. the ability to compute.
14. the ability to reason/infer.

In addition, we might divide the digital actors in different classes, which could be defined in terms of the above identified abilities:

1. Sensors: Actors which can take measures in the physical and/or digital world.
2. Manipulators: Actors which can manipulate objects in the physical and/or digital world.
3. Tools: Actors which can conduct well-defined basic computational work (typical applications, storage, etc)
4. Translators: Actors which act as translators to enable the communication between other digital actors.
5. Bridgers: Actors which can translate between actors in the physical world and actors in the digital world (a voice interface, a GUI, or a brain interface).
6. Brokers: Actors who known about the ability of other actors (in the digital and/or physical world) and are able to bring them together to collaborate.
7. Agents: Actors who are able to execute complex tasks involving reasoning and trade-offs. They are likely to do so on behalf of a human actor in the physical world.

Now look at some basic examples from IT through the perspective of this paradigm.

1. SoA: A service is an ability on offer by a specific actor. They may be advertised by a broker.
2. Word processor: Is a tool using a bridger (the GUI) to obtain input from a human actor telling it what to do in manipulating another digital actor: the document.
3. Navigation system: Is a tool which talks to a sensor telling it where we are, and using a bridger to obtain from the driver where they want to go, as well as telling the driver what to do.
4. Digital camera: Is a tool taking a picture on a command issued by way of a bridger (the release button) and consequently talking to a storage tool (materialized as a the memory card) to store the picture. At home I physically move the memory card housing the digital actor "storage tool" to my computer's card reader and tell another tool (e.g. Adobe's Lightroom) to fetch it.
5. Screen scraping: The worst kind of interaction between digital actors, where the two actors need to resort to using two bridgers to interact.

Consider, as a more elaborate example, my address book. My address book really is a collection of digital actors (tools). It is not a single tool actor, but really a collective: some of them live on my notebook (Thunderbird's address book and Outlook's address book), one lives somewhere out there (Plaxo!), another one lives on my palm, and one living in my phone. They keep each other in-sync using translators. Now if we look at reality, we immediately see some potential improvements of this scenario, and the technologies used:

1. Regretfully in some cases I have to initiate the synchronisation manually. What I would love is for them to keep each other in-sync automatically. When my phone is able to contact the “big” digital ecology (the Internet), it should sync with the address book living at Plaxo automatically.
2. Why do I need two address books on my laptop? Can’t both tools (Thunderbird and outlook) talk to the same address-book-tool?
3. When I buy a new phone, I need to look for a translator myself to enable synchronisation between my new phone and my pre-existing collective.

As a human user I am willing to accept that I must be aware of the fact that I need a collective of tool actors to create my address book, since the physical dependence of the digital ecology might not always allow for an interaction between my phoning-tool and my address-book-tool. However, why do I need to be bothered by manual synchronisation, and even worse, translations during synchronisation. This makes the role of the translators important. Ideally, when I’ve bought a new phone, I would like to tell my address-book collective: listen guys, this is Nokius, he speaks Finish, now start talking to him. They would then invite the right translator to join their collective and then update Nokius, and keep it in sync.

So how do we create this world? First of all, it is already possible to use a paradigm like the one discussed here as a design paradigm for systems. In other words, when designing new systems we might be looking for the different types of actors (and most probably be adding new types to the paradigm). Furthermore, by tuning the behaviour of actors in the web, in particular the agents, to the needs of the people, we can achieve a more personalsed experience. And beyond that? What about a societal experience? Imagine the actors in the digital world and the physical world being so well integrated that we work together as a collective. Sounds funny? Imagine an agent from the digital world, who/which has been doing a number of useful things for you, asks you to do something in return because it is working on a task for someone else ...

Posted 12th February 2008 by Erik Proper

3 View comments

erics 12/2/08 05:34

"So how do we create this world?"

Man am I glad I don't have to build your POC's anymore! ;-)

Reply

LouisD 12/2/08 10:45

Great concept Erik! It triggered me to think about the vast opportunities that can arise from this ActorWeb concept. I envisage a future world where actors will exist in more flavors: self contained human actors, self contained digital actors or hybrids (humans with integrated digital actors "on-board", for example carrying a pacemaker). These actors could increasingly talk a common "services" oriented language to eachother, of course constrained by attainable security measures.

The design of a digital actor could follow the service oriented paradigm and thus should be able to expose its capabilities (through it's service interface) to it's environment. In my opinion, an actor should be as self contained as possible and thus should have it's own services advertising mechanism on-board. It should not need to rely on a central broker (except maybe for authentication purposes). This paradigm could become the base for a self-regulating, self-learning network of actors. The only thing actors would need in common is a (rich, extendable) open standard language and enough guarantees for actor network stability and reliability. By the way, Professor Maarten van Steen (Vrije Universiteit) performed some very interesting research in this area.

Let's take an example of a digital camera that exposes itself. It could be in a passive or active communication modus. In passive mode, it can wait until it gets queried by another actor in it's environment. In active mode it could for instance be sending out a service advertising message something like "i am currently available and can offer a set of public services - please connect to me to find out what services i can deliver or compose for you and what the cost of these services is". The actor in the environment that discovered this digital camera can now decide to connect and query what kind of services the camera delivers. The actors might even construct composite services "on the fly" (or call in the assistance of third party actors) or otherwise extend each others service portfolios. Endless possibilities!

Maybe the greatest challenge will be to invent a common actor language that is open, rich, extendable, reliable and fully adopted by the industry.

Reply

Jos Groenewegen 26/4/08 16:34

A very interesting concept, and most definetly something to strive towards. Seeing the digital architect at work, being a student in information science (gosh:P) I can't really say I'd have anything against this.

Still seeing the request at the beginning asking for discussion I'd say it's worth tossing an idea out there.

The whole digital architect as demonstrated here is still limited in putting in new connections, however he doesn't add new variables with the sole purpose to make for a better link.

When looking at the different types of actors wouldn't it be interesting to consider actors in the digital world , (regardless of current existence), with the goal not to manipulate objects in the digital world but instead to keep data that had in the past been bridged present. Say, storing information about people's feelings when reading a text. Allowing the ability to look at meta data collected by bridgers?

Just a way to possibly bring things such as data-mining to a higher level, but more importantly to give a digital architect a far broader range of tools. One of the great difficulties with architecture I'd say is not merely getting everything to interact but is to overlay the complex humans and their desires onto a digital architecture so it knows how to respond at a given time as well.

Adding meta data regarding bridgers to the repetoire of abilities of the digital architect might be a step to making that more viable.

The most important thing remains to map the world in such a way that complexify can be brought back. But to me the real challenge lies in finding the wish in the right way, ensuring the architecture can find that and move from there.

When creating a new world we'll be looking at the behaviour of actors, agents, but as you say yourself at the end the personalized experience is important.

So a focus on finding what is requiered for that personalized experience is essential.

Random mussings of a student late at night!

Reply

Enter your comment...

Comment as: Google Account

Publish | Preview