Agent-based development

Description

This document is an essay about our perception of agent-based softwares now and in the future. It was written for the first group assignment of the **DBA** (*Desarrollo basado en Agentes*) course in *Universidad de Granada (E.T.S. de Ingenierías y de Telecomunicación)*, and submitted on **October 13th, 2015**.

Group members

- James NOLAN
- Fernando SUÁREZ
- Zacarías ROMERO
- ~~jaordud~~
- ~~Sergio LÓPEZ~~
- ~~German GARCIA~~

Discussion

Subject

This essays aims to answer the following questions:

> What do you think are software agents today and what do you believe they will be in the next 10 years?

We will answer to both these questions in a two-parts essay:

i. In the first part, we will try to express what we believe software agents are while providing analogies and examples.
ii. In the second part, we will speculate about what software agents might be in the future.

In the conclusion, we will discuss what kind of potential issues this evolution might raise.

Agent softwares today

The way the question is formulated lets use believe that we should rather explain our own personal perception of what software agents are rather than copying definitions found in Wikipedia.

At the time of writing this text, agent-based softwares is a new subject for us.

We probably already have a superficial knowledge of what they might be because it

would be foolish not to understand a course title or do a quick research on the subject before applying to it, but we cannot claim to have a clear perception on the matter. Therefore, we are going to try to give a first *intuitive* explanation on what software agents are.

Our understanding is that a software *agent* is an automated system (as any other software), part of a greater “super-system” or “global system”, that has the extra responsibility to make *individual* decisions based on its interpretation of other agents’ behaviour and act accordingly in order to allow the super-system to fulfill its purpose.

This task delegation mecanism would allow to decentralise organisation in order to solve problems whose nature make it difficult to tackle in a *monolithic* fashion.

We can think of many analogies of that paradigm in nature. For example, the following image is a screenshot of a timelapse movie of a hundreds of white blood cells neutralizing a worm. The collaboration of white cells help the immune system to fulfill its task. The full animation is available here.

Here, the *global system* is the immune system. It’s *purpose* is to eliminate a harmful organism. White blood cells can be seen as *agents* that *collaborate* by analysing and reacting to other components’ *behaviour* to allow the global system to reach its goal.

With this analogy, we hope to illustrate our understanding of agent-based problem solving when the issue has favourable properties. When it comes to software engineering, there are many applications where the same kind of organisation can help us solving problems.

For example, many searching, sorting or pathfinding algorithms can be parallelized into multiple threads to take benefit of multi-cores architecures and thus improving overall speed. Each thread can be seen as an simple task agent that has to observe the state of other threads to decide what it has to do (continue, stop, wait, move to next range…).

It can also be useful to decompose software into agents when we need to emulate real life situations involving many independant components such as crowds (group of persons), trafic (group of cars), fluid dynamics (group of liquid particles). Those simulations allows us to avoid crowd crushes, understand traffic jams or rendering floods. In those cases, agent based modeling is more practical because it appears more natural to assign behaviours to components than to the global system, which is often impossible because of its chaotic nature.

Agent softwares In the future

Now that we have stated our understanding of agent based softwares, we will try to imagine what might change about them in the next ten years. Of course, we assume we are allowed to be a little more creative in this part.

Technologies are often initialy developed in a safe environement with a limited impact in case of errors. But frequently, if they they happen to be cheap, useful for the industry or the army, they tend to be used in more critical projects with higher responsibilities. Since hardware is getting cheaper everyday, and since bio-inspired technologies are subject to many research for industrial applications, we believe it might happen with software agents development.

In our opinion, until now, software agents are extremly helpful to:
- Emulate real life phenomenon like bird flocks,
- Building parallel and scalable systems such as sorting algorithms or logging systems,
- Organizing moving hardware components like robots,
- …

One common factor in these tasks is that agents have very small responsibilities. If a water particle from a flood simulation doesn't behave correctly, it will probably not affect the global system in a catastrophic way. If a logging server crashes, it may affect latency but it can be easily repaired to set the global system back in an optimal configuration. If a small robot is stuck in a wall, we might have to put it back on track manually but it won't harm anybody.

In the future, software agents might be given heavier responsibilities. For instance, self driving cars such as *Google driverless cars* might become available to the general public as soon as 2017 and some experts predict that wide usage will induce traffic congestion reductions. For this to happen, self driving cars will have to behave as agents to collaborate in an attempt to reduce the "stop-and-go" motion.

Soon, *military* grade drones might fly as flocks, making a single programming error much

more harmful and costly than a small quadricopter. Robots might help medics retrieving injured people in inaccessible locations after earthquakes, avalanches or floods.

Nanotechologies could also take advantage of agent based development, helping surgeons perform difficult operations.

Conclusion

These highly sensitive responsibilities, of course, raises concerns from a legal, ethical point of view just as much as any other “new” technologies. Who will be charged in the case of a deadly software failure? How will people cope with enthrusting machines with their health?

But one major concern that applies even more to software agents is **security**. Only in the past few years has security been a real concern for developers. The agent based paradigm heavily relies on remote communication to ensure a proper collaboration. If communications can be cut, falsified or altered, the whole system cannot fulfill its purpose and could be even be used in a malicious way. It is therefore essential for developers to keep aware of it, to ensure a high level of security in such systems.