Pitfalls of Agent Development

José M. Vidal

Tue Nov 11 12:36:57 EST 2003

We summarize

- Michael Wooldridge. Introduction to MultiAgent Systems.[1] John Wiley and Sons;. 2002. Chapter 10

1 Categories

- We are now talking about building multiagent systems for industry.
- The pitfalls fall into one of seven categories.

1. Political
2. Management
3. Conceptual
4. Analysis and design
5. Agent-level
6. Society-level
7. Implementation

2 You Oversell Agents

- Agent systems will only provide a benefit on specific instances.
- Look for systems that need to be distributed (robustness, security, performance).
- Look for systems that need to be open.
- Look for systems where no satisfactory centralized solution exists.
- Try to get by with as little AI as possible.

[1] http://www.amazon.com/exec/obidos/ASIN/047149691X/multiagentcom/

3 You Get Religious

- Agents are not a religion, just a tool.
- They are not the solution to every problem.
- They will not make you hip.

4 You Don't Know Why You Want Agents

- Understand why you want agents.
- Research.
- Improved X
- No other technique will address the problem.

5 Provide Generic Solutions To One-Time Problems

- Do not build another agent testbed/simulator just because you can.
- There are plenty of systems: FIPA-OS, JADE, ZEUS, NetLogo, etc.
- You will probably not be able to re-use something if you built it for a specific applications (no matter how "generalized" you think you are being).
- It takes a lot of effort to make something generalized.
- One of the tenets of extreme programming[2] is not to over-design. Build only what you need now.

6 Confuse Prototypes with Systems

- Prototypes are easy to build.
- They are not systems.
- Going from one to many machines is very hard.
- Many machines on different sites is even harder.
- Many machines on different OSs is impossible ;-)

[2] `http://www.extremeprogramming.org/`

7 Confuse Buzzwords with Concepts

- The word "agent" can have many different meanings.
- People often think they understand what an agent is, when they really don't.
- Make sure all the words you use in your design have a specific meaning. That is, they should be nouns, not adjectives.
- A *"robust secure distributed synergistic system"*
  could be anything.

8 Forget That Agents Are Software

- Agent-based software engineering is still software engineering.
- Agent systems are new, so we do not know too much about how to build them.
- Do not focus solely on the agent parts, there are also databases, user interfaces, deployment issues, etc. to take care of.
- You still need requirements analysis, specification, design, verification, and testing.

9 Forget It is Distributed

- Distributed systems are very tricky to design an implement.
- People have a hard time thinking about asynchronous interactions.
- There are many complex technologies available: CORBA, RMI, SOAP, etc.
- In fact, distributed programming is a whole other class (CSCE 590).
- Agent systems build on top of distributed programming (OSI stack).
- Make sure you use existing technologies.

10 Don't Exploit Concurrency

- A bad multiagent system is one were everything has to be done in sequence.
- Your first job is to determine how the work can be split into smaller parts and which of those parts can be done concurrently.
- If there is very little concurrency then maybe agents is not the best solution.

11 Want Your Own Architecture

- There are many architectures out there (subsumption, BDI, layered, etc.). Start with those.
- This need is usually driven by the "not designed here" mentality.
- Developing an architecture is a long progress, and you will still not know if it actually works!
- Use something that has been tested over the years, then try to improve it.

12 Believe Your Architecture is Generic

- If you did develop an architecture you might think it is generic. It is not.
- Different architectures are good for different types of problems.
- Determine what type of problems your architecture is good for.
- Try to understand why your architecture works well for your problem.

13 Use Too Much AI

- There are a lot of AI techniques available: machine learning, planning, speech recognition, etc.
- Only use those techniques that actually help to solve your problem.
- Do not believe that your agents have to be smart.
- Remember that people are already smart. Sometimes agents can exploit this.

14 Use Too Little AI

- The agents need to show some form of coordination.
- If there is no decision-making then there is probably no need to call it an agent.
- Overusing the term "agent" will lead to its losing its meaning.

15 See Agents Everywhere

- You need to choose the right grain size
- What are the agents in your system? Think carefully about this.
- Try out different agent instantiations to see what emerges.
- If you have too many agents you might end up with undesired emergent behaviors. Be careful. Keep interactions to a minimum. Keep protocols simple.
- If you have too few agents you will end up placing all the functionality on a few.

16 System is Anarchic

- You cannot just throw a bunch of agents together.
- Define appropriate interaction rules (mechanism design).
- Provide an organizational structure.

17 Confuse Simulated with Real Parallelism

- It is hard to make a multi-threaded program into a distributed program.
- There might be hidden sharing of information.
- Timing issues.
- Timeout and communication problems.
- Load balancing problems.

18 Ignore Standards

- Use existing de facto standards.
- FIPA
- CORBA
- RMI
- SOAP
- XML
- RDF

- DAML
- DAML-S
- OWL

This talk is available at `http://jmvidal.cse.sc.edu/talks/agentdevpitfalls`