Reporting about the Mod Software Process

Giovanni A. Cignoni

Dipartimento di Informatica, Università di Pisa
Corso Italia, 40 Pisa, Italy
giovanni@di.unipi.it

Abstract. This short paper reports about the software process used for the development of *CFS2 mods*. CFS2 is the latest version of MS Combat Flight Simulator; a mod is a modification of the game that adds new features, fixes existing bugs, or enhances its performance. Mods are not developed by MS; indeed there is a wide community of enthusiastic fans of CFS2 – and smart programmers – that provide free mods through the Internet. This community is a completely anarchic one, but, if we carefully observe it, we discover a repeatable software process. The paper provides an inside report of such process and proposes several discussion issues about the differences between this particular context and the more known industrial one.

1. The Context

Since its early versions, *MS Flight Simulator* was implemented as a very open system. Developing add-ons for the game – many will disagree with this definition – has always been a consistent part of the entertainment. *Aircrafts* can be modified in their look and performance, even created from scratch. The same is for *scenarios* (i.e. terrain representations, airports, and cities) and *situations* (things like landing with a damaged engine in a windy day on a short runway, ice probable).

The more than ten-years history of the game counts 7 major releases of the simulator engine with two intermediate releases, 6.5 and 7.5 devoted respectively to *MS Combat Flight Simulator 1* and *2*, the editions that add the fight experience to the game. The last version, CFS2, has a very impressive number of features that can be modified: besides aircraft and scenarios there are *ships* (CFS2 is specifically dedicated to the Pacific Theatre of Operation in World War II) and *ground vehicles*. Aircrafts and all other objects have *weapon* and *damage* profiles. Situations are replaced by far more complex *missions* and *campaigns*. Finally, there are *effects* used to model smokes, explosions, ship wakes and many other visual issues.

From a technical point of view, developing a mod means writing code and integrating it in a new configuration of the CFS2 system. Many different programming languages are involved; here are just some examples. Object as buildings or vehicles are modelled through a procedural 3D language, sources have to be compiled and integrated so than the simulator engine can execute them to actually draw the objects. Effects are programmed as particle systems defined through sets of declarative statements. The dynamic part of a mission is programmed through a pre/post condition language that uses triggers and actions.

V. Ambriola (Ed.): EWSPT 2001, LNCS 2077, pp. 242–245, 2001.

In some cases there are visual tools that help the development process, as, for instance, the *Mission Builder* bundled with the game or the commercial object designers available by third part companies. However, hacking the source code is a common activity, preferred by developers who want to have access to all system features, in particular to those not documented.

Support from MS is poor: the game was shipped in late October 2000 with many bugs; official documentation about the simulator engine is practically non-existent or released sporadically [1]. On the other side, in three months, the mod developer community fixed several important bugs (just to cite two: the incoherent behaviour of some ships and the ghost payload on add-on aircrafts) and provided many new features (seaplanes and ground troops, as two representative examples). The total number of mods available – as freeware – on the Internet is simply uncountable. In the same time, MS did not release any patch or upgrade. One possible explanation of this decision is that MS do not want to deprive people of such entertainment.

Internet plays a central role in this context. The community of mod developers and the wider community of game players (i.e. mod users) live and communicate on the network. Basically, two fundamental services are needed:

- *public repositories*, where developers publish their new mods and users can download them; repositories are generally controlled and content reviewed to assure the quality of the available mods;
- *forums*, where user have news about ongoing projects and receive support; moreover, dedicated forums are used by developers for technical discussions and information exchange; forums are usually moderated.

Several Web sites provide such services. Two of the most representative examples are [2, 3], but many others exist. Access is generally filtered by some requested membership, that, while offered for free, is used to maintain a little order in the community. Besides these loose control mechanisms, all is left to self-regulation.

Developers are very collaborative. Sharing information and results is a general attitude. In many cases help and specific suggestions are directly provided. Joint projects are common and carried out by developers that co-operate using forums and e-mail as co-ordination means.

2. The Process

Building CFS2 mods is anything but software development. Users and developers are involved, tools are used, and software products are released. Observing the way mods are ideated, implemented and released, it is possible to figure out a process that generally goes through the following phases:

1. ***inception***; have the idea, draw a first draft of the mod requirements;
2. ***experimentation***; be sure of the mod feasibility; carry out experiments, discuss technical issues through forums;
3. ***design***; define how the mod has to be implemented, which files have to be added or modified, which solutions have to be used;
4. ***code***; write down the first version of the mod files; also build the test environment, generally one or more missions to test the mod;

5. ***test & debug***; install the mod on your system, test, rework, test, rework, system crash, test, rework, ... eventually the mod will work;
6. ***test & tune***; same as above, but the goal is to fine tune the look and the behavior of the mod; less system crashes, hopefully;
7. ***packaging***; write the documentation, build the distribution;
8. ***final test***; test the distribution, check that documentation is correct;
9. ***deliver***; upload all, report to the forums, mission accomplished!

The mod software process is a rather classic waterfall with three phases, 2, 5, and 6, that internally are highly cyclic and evolutionary. There is a deep distinction between phases 5 and 6 because exiting phase 5 is the true turning point of a project. The whole process is iterated to build a new release of the mod – a mod of the mod.

There are cases in which the process is truly defined, as it is for the *Assembly Line Process* [4], a process devoted to recreate flight models that have less than 1% tolerance with respect to the real specifications of the aircrafts – far more accurate than the stock aircrafts provided with the game. This Cleanroom-like process absolutely forbids the evolutionary approach and provides spreadsheets to check the correspondence of code to specifications.

More generally, the process is not stated or formalized. Many developers, considering themselves artists or historians more than programmers, if directly asked, will deny their connection with a process. However, the process can be observed and considered as a consolidated praxis. In a CMM-like evaluation it will rate something less than level 2: the necessary self-discipline is in place to repeat the process on similar projects, but schedule and costs are not tracked – well, it's a hobby!

3. Some Discussion Issues

Observing such a particular context (and analysing it under the software process perspective) points out several discussion issues. All of them relate with differences between the anarchic community of mod developers and the – supposed to be – organised commercial software production. It is worth to note that we are talking of a hobby. The performance of the process is out of the debate: it is not possible to compare people developing for the joy of it with people doing that for work.

The first issue is about process definition. Many experiences report about difficulties in the adoption of defined development processes in the software industry. For instance, a survey on 36 enterprises in Centre Italy [5] showed a dramatic situation: 84% of them do not reach ISO 15504 level 1, and 57% never tried to define their development process, even informally. The close observation of a community of hobby developers shows that a basic software process is in place.

The second issue is about the motivations. Process definition is a way to communicate with the customers. This is a goal stated by many standards: from ISO 9001 to ISO 15504, the process is a way to show how the developers will work, as a quality assurance issue, but also as a way to better reporting about project status. To a certain extent, mod developers and mod users are aware of the underlying development process and use it to communicate about advances in the projects.

The third issue is about configuration management, a classic topic in the software process, a particularly crucial one in a context where hundreds of people modify a

system that counts more than five thousands files. While it is difficult to have configuration standards applied in the mod community, the need and the willing to have some control are clearly stated. For instance, almost all developers consistently keep versions of their products. The same need is not perceived in the industry, where often configuration is an obscure or misunderstood term.

It is normal that mod developers put a lot of effort in their hobby: they enjoy it. Such effort justifies their successes: maybe they are not smart programmers, they just do many trials. So we cannot conclude that their results derive from their process. But they use versions, ask for configuration management, have a process and use it to communicate with their users. Compared with the commercial software production, this awareness of best practices is an unexpected result.

Acknowledgements. The community of mod developers and users made possible these considerations. I also bored them in the forums with process related questions. Vincenzo Ambriola suggested writing the paper and helped in reviewing it.

Flight Simulator and Combat Flight Simulator are commercial trademarks of Microsoft Corp.

References

1. *MS Combat Flight Simulator*, official site, http://www.microsoft.com/games/combatfs/.
2. *Combat Flight Center*, public repository and forums, http://www.combatfs.com/.
3. *CFS2 Online!*, public repository and forums, http://cfs2.dogfighter.com/.
4. *Flight Model Research Institute*, Assembly Line Process, http://home.socal.rr.com/flighttest/.
5. G. Bucci, M. Campanai, G.A. Cignoni, “Rapid Assessment to solicit Process Improvement in SMEs”, Proc. of the *EuroSPI ’2000 Conference*, Copenhagen, November 7-9, 2000.