AA: the Algorithmic Auto-regulation (Software Development) Methodology

Renato Fabbri Ricardo Fabbri Vilson Vieira
Daniel Penalva Alexandre Negrão Lucas Zambianchi
Marcos Mendonca Danilo Shiga

June 1, 2013

Abstract

We present a new methodology for coordinating teamwork called Algorithmic Auto-regulation (AA). This methodology is based on recent social networking concepts and individual merit and exhibits convenient asynchronous, auto-documenting and auto-regulating characteristics. Team members take on an egalitarian and voluntary role, by logging into periodic "AA sessions" for an arbitrary duration (e.g. 2 hours per day). During each session, a user creates a log composed of short text sentences about their activity. These logs are publicly aggregated in a Website and are peer-validated, as in code review. A short screen-cast is ideally recorded at the end of each session to make AA logs more understandable. This methodology seems to be well-suited for increasing the efficiency of teams working on Global Software Development (GSD). Recognized reasons for this are the: 1) built-in asynchronous on-demand communication, documentation and proven working hours; 2) reduced need for central management, meetings and time-consuming reports. Hence, AA legitimizes and eases activities of a distributed software team. It enables groups to have new means to fund activities, allowing for business models to emerge from distributed software development. AA is proposed, at it's core, as a way of having self-replicating hacker initiatives. These claims are discussed in a real case-study of a distributed free software hacker team called Lab Macambira.

1 Introduction

One of the defining features of modern times is the widening geographical distribution of software teams [8]. This is responsible for the so called Global Software Development (GSD) [6, 1, 2]. An example is the free software movement. Projects and institutions like Mozilla Foundation has several employees and thousands of voluntary developers distributed across many countries. The same is true for GNOME [6], OpenBSD, MySQL or Apache Software Foundation, to cite just a few[1]. Along the free and open software projects, GSD has a growing popularity in every niche of the software industry as a whole, even on those distributing their software with proprietary licenses. This phenomenon is attributed to a variety of factors such as a larger labor pool, natural globalization of software companies and foundations or even the premise of cheaper cost of production [7].

Despite the advantages of GSD, it is known how difficult it is to coordinate and fund free software on a larger scale than currently available. That is, when teams are very heterogeneous, containing not only volunteers and experienced developers, but also contractors from different backgrounds and cultures. This observations are in consonance with dedicated studies, such as [4]. As a common base, related difficulties for GSD gather around distance, time and cultural differences. In the case of free or open software projects, all these factors are involved.

Another problem faced by modern software companies and other collectives are frequent ineffective meetings, which are seldom focused on the interest of any attendant. The result is that it has become the norm to participate in these with the "laptop open", which is unproductive. Software developers like to code, to be productive, to have their hands on their project, to do what they are best at. They dislike to have to stop for meetings or to write lengthy reports to justify their funding.

To address these matters is the purpose of AA, a methodology and an associated software system for coordinating distributed team work dealing with the disadvantages of GSD. Team members take on an egalitarian role, and stay voluntarily *logged* in the system for part of their time (e.g. 2 hours per day), during which they log a periodical short text sentence — similar to a "tweet" in this Twitter era — as the status of their activity using an easy to use command. These "micro-blog sentences" are publicly aggregated and validated by other team members. Through AA, the community has a methodology and an associated system to validate and enable the activities of a distributed software team. It implicitly legitimizes financial support for the expansion of the activity of the developer team as the activities are documented and validated. The AA methodology is specially useful for coordinating distributed and decentralized team work, providing effective means to asynchronously update different team members, diminishing the need for harsh synchronous meetings.

A brief overview of current papers about GSD methodologies related with AA is presented in section 2. In section 3, the most relevant characteristics of the AA methodology are outlined. In this same section, there is a description of an AA use in a team of 9 developers in the second half of 2011 and a broader a more released use of AA from 2012 until now. Finally, in Section 5, there are final conclusions and indicatives of future possibilities to the practical use of AA on other teams of software developers or individuals working on non-software

[1]Ohloh, the open source network, have a more complete and constantly updated list of the most active projects on-line at `www.ohloh.net`

activities.

2 Related work

There has been a large amount of research in methodologies to deal with distributed teams of developers. Although this article focuses in GSD, some principles involved on those methodologies could be used on smaller teams of developers working in the same place, time period and with minor cultural differences. Moreover, "distributed development" is generally tough as being global which is not true. AA was applyed to a team that work at the same city but on different times (more details on Section 4). Even smaller groups of developers working on the same building could use GSD methodologies. A thorough survey of these methodologies is beyond the scope of this paper. This section presents a brief overview.

Various methodologies for GSD were built around the factors that affect distributed team works. As proposed by Carmel [4], these comprises three distances: geographical, cultural and temporal. Geographical distance prejudice *coordination*, being the act of integrating all the tasks distributed between units [5]; *control*, or the process to maintain specific goals, policies or quality levels; and *communication*. All these factors are correlated, for example, a team needs to have good communication to work on tasks related to a specific problem. Cultural distance encompass differences on organizational and natural culture. Spoken language, unit and ethnic values are common forms of this distance. Some companies prefer to situate development units in foreign locations with minimal cultural distance (e.g. an American firm prefer Ireland, because of spoken language similarity [5]). Finally, the temporal distance prejudices synchronous communications like telephone or video-conferences. Developers working on different time-zones manage their agenda guided by this temporal distance.

Targeting geographical distance, Carmel [5] suggests a tactic to enhance intensive collaboration. His approach divides the whole software life-cycle on levels of complexity. Each level has a degree of collaboration. For example, some developers working on a project with high collaboration level should use the follow-the-sun approach: when concluding the work day, they pass their work to the team working on another time-zone. Other tactics are suggested by the same author to deal with the three distances, like separating foreign units of developers in time-zone bands.

Battin et al. [3] argues about their experiments using methodologies created for the development centers of the Motorola Company (which has over 25 software development centers worldwide). These methodologies included constant communication with critical units, incremental integration and schedules based on time-zones distributed to developers on 6 countries from 3 continents.

While considering free software projects instead of companies, similar factors are present and some methodologies arises. German [6] gives a concise review of methodologies used by the GNOME project, one of the most active free

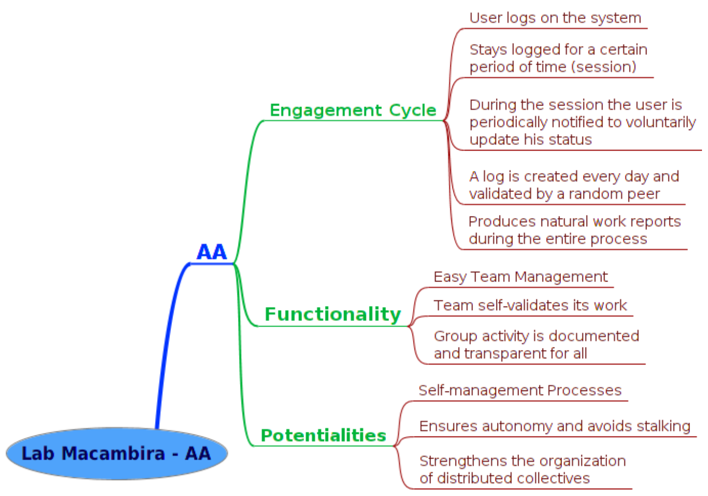

Lab Macambira - AA
AA
Engagement Cycle
User logs on the system
Stays logged for a certain period of time (session)
During the session the user is periodically notified to voluntarily update his status
A log is created every day and validated by a random peer
Produces natural work reports during the entire process
Functionality
Easy Team Management
Team self-validates its work
Group activity is documented and transparent for all
Potentialities
Self-management Processes
Ensures autonomy and avoids stalking
Strengthens the organization of distributed collectives

Figure 1: Mindmap of AA methodology: 1) developer engagement cycle, i.e. his use of AA; 2) functionality as main goals of the system; 3) potentialities as enhancements AA delivers to the developer's medium.

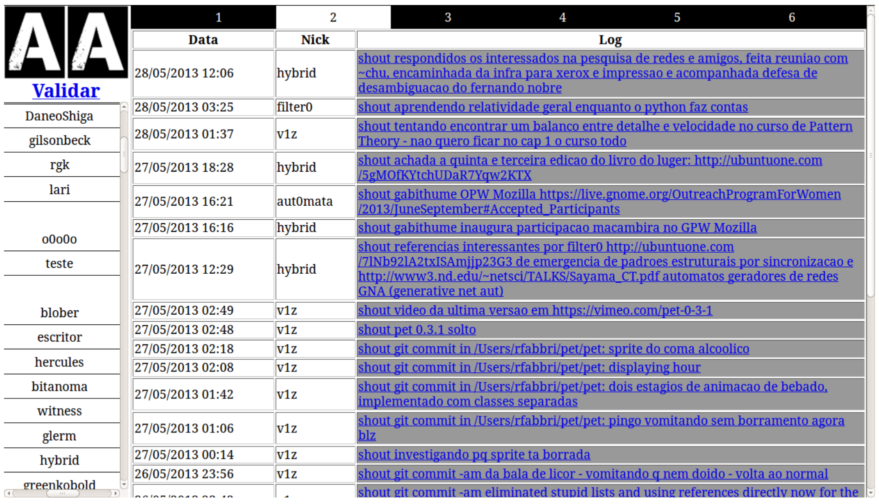
software projects and used by companies like Sun Microsystems. It is interesting to note a difference on the viewpoint present on his paper: German focuses the methodology description on code. He start explaining that GNOME is separated into modules (76 on version 2.4, to be precise) and each module has one maintainer who divide his modules into separated parts in which other developers can work on independent tasks, along other responsibilities. As like others free software projects, all the development was done using a *bugtracker* to bugs and issues management, mail lists and Internet Relay Chat (IRC) to discussion and communication and a software version management like SVN or Git. He finishes by observing that periodical, e.g. yearly, conferences like GUADEC is common on free and open source projects for face-to-face meetings and is based each time on a different place.

3 The AA methodology

As noted, some strategies for GSD is based on complex methodologies and many of those were built for a specific company or software center. This section proposes an alternative based on a simple idea: small working sessions logged by a computational tool. Figure 1 summarizes this methodology.

3.1 The AA session

From a developer's perspective, the AA methodology is based on creating small, high perspective, reports of what is being done in a specific time frame. This



Data	Nick	Log
28/05/2013 12:06	hybrid	shout respondidos os interessados na pesquisa de redes e amigos, feita reuniao com ~chu, encaminhada da infra para xerox e impressao e acompanhada defesa de desambiguacao do fernando nobre
28/05/2013 03:25	filter0	shout aprendendo relatividade geral enquanto o python faz contas
28/05/2013 01:37	v1z	shout tentando encontrar um balanco entre detalhe e velocidade no curso de Pattern Theory - nao quero ficar no cap 1 o curso todo
27/05/2013 18:28	hybrid	shout achada a quinta e terceira edicao do livro do luger: http://ubuntuone.com/5gMOfKYtchUDaR7Yqw2KTX
27/05/2013 16:21	aut0mata	shout gabithume OPW Mozilla https://live.gnome.org/OutreachProgramForWomen/2013/JuneSeptember#Accepted_Participants
27/05/2013 16:16	hybrid	shout gabithume inaugura participacao macambira no GPW Mozilla
27/05/2013 12:29	hybrid	shout referencias interessantes por filter0 http://ubuntuone.com/7lNb92lA2txISAmjjp23G3 de emergencia de padroes estruturais por sincronizacao e http://www3.nd.edu/~netsci/TALKS/Sayama_CT.pdf automatos geradores de redes GNA (generative net aut)
27/05/2013 02:49	v1z	shout video da ultima versao em https://vimeo.com/pet-0-3-1
27/05/2013 02:48	v1z	shout pet 0.3.1 solto
27/05/2013 02:18	v1z	shout git commit in /Users/rfabbri/pet/pet: sprite do coma alcoolico
27/05/2013 02:08	v1z	shout git commit in /Users/rfabbri/pet/pet: displaying hour
27/05/2013 01:42	v1z	shout git commit in /Users/rfabbri/pet/pet: dois estagios de animacao de bebado, implementado com classes separadas
27/05/2013 01:06	v1z	shout git commit in /Users/rfabbri/pet/pet: pingo vomitando sem borramento agora blz
27/05/2013 00:14	v1z	shout investigando pq sprite ta borrada
26/05/2013 23:56	v1z	shout git commit -am da bala de licor - vomitando q nem doido - volta ao normal
[illegible]	[illegible]	shout git commit -am eliminated stupid lists and using references directly now for the

Figure 2: AA Version 0.1. Messages of users: hybrid, filter0, v1z e aut0mata about activities of interest for labMacambira.sf.net and other parties (IPRJ/UERJ, IFT/UNESP, IFSC/USP, OPW/Mozilla, PulaPirataComics). Each message is a "shout", which, grouped, forms an AA session.

frame can be something between 5 to 15 minutes, depending on what is comfortable for the developer. An AA Session is a period of about 2 hours, in which these short reports (called "shouts") are periodically written. The developer can set reminders to show up when it is time to make a new report. The objective of the flexible time frame and reminders is to minimize developer overhead during his AA session. In this way, he can make the reports with little or no loss of concentration on his coding activity. Each report can be sent directly to an online server, or stored locally in a temporary database for sending later on. This makes it possible to use AA while offline.

Developers can also record a video screencast at the end of the session, summarizing what has been done, explaining with his words and showing on the screen his most important results. This, combined with the textual log of his session, makes the whole report more significant as documentation and understandable for himself or other people searching for information about his production.

3.2 The AA website report

All AA reports made by the developers are sent to a Web server and become public in a Website. It is then possible for a manager or another developer to follow closely the work of a developer, nearly real-time, reading each of his small reports of what he is doing.

Another possibility is to check older sessions to see what was done and the comments of the developer about it. Since each AA post (or “shout”) happens in a small time frame of work, the information about what was done becomes easy to understand, which seems better than a long report in the end of a session.

The developer can add to his session a link for his screencast about that session. This screencast is a video summary of what was done and complements the report.

3.3 Validation

Each AA session must be validated by another developer. This means that all reports are read by someone that will consider valid or not and will even write comments about the specific session. The developer in charge of validating a session is decided randomly by the AA Web server, which sends an email to the chosen developer with the URL to a validation interface.

4 Results and discussion

The easy and effective management of teams working on GSD is the main purpose of the AA methodology. This methodology was applied to a group of 9 developers in the second half of 2011 on what was named “Lab Macambira”[2]. The main objective of the team was to work on different free software projects, contributing directly to their development, sending bug corrections or proposing new features on their source code.

Team members earned knowledge on software development as part of large and distributed free software projects like Scilab and Mozilla. In this way, one month of training was conducted by three experienced developers, teaching the basics of development support tools like bugtrackers, programming languages and version control systems. After this period, a challenge was proposed for the new developers: send a bug correction or a new feature to a large free software project and you will work with us, you will be a “Macambira” developer. Table 1 summarizes the contributions of each “Macambira” to free software projects.

In one month, “Macambira” developers contributed to many large free software projects. Some of the developers started the training with no knowledge of what is free software and ended that period becoming a free software developer.

During that month, the same team developed the first version of the AA system and used AA to manage their activities, even while developing the system. All the source code of AA — the client that sends the logs and the AA Web server — is publicly available[3] and all the AA session logs of the “Lab Macambira” team is also online[4].

After the training period, during 6 more months, the “Macambiras” worked on a large range of free software projects, distributed on work groups. Each work

[2]LabMacambira.sf.net: http://labmacambira.sf.net

[3]AA source code: http://labmacambira.git.sourceforge.net/git/gitweb.cgi?p=labmacambira/aa

[4]Logs of AA sessions: http://labmacambira.git.sourceforge.net/git/gitweb.cgi?p=labmacambira/paainel

Table 1: Free and open software projects that received contributions from "Macambiras". On the first column there is a list of applications. At right, the pseudonym of the "Macambiras" who sent *commits* to the application. At "Lab Macambira", and at free software community in general, is a common practice to use a pseudonym as identification.

Application	"Commiters"
Mozilla Firefox	daneoshiga, bzum
Evince	hick209, bzum, marcicano, mquasar
BePDF / Xpdf	marcicano
Ekiga	flecha
Empathy	fefo
Lib Folks (Telepathy)	kamiarc
Scilab	v1z, humannoise
VxL	v1z
ImageMagick	v1z
OpenOffice	hick209
Puredata	v1z, automata, greenkobold, gilson, bzum
Puredata OpenCV	v1z
Puredata GEM	v1z, fefo, hick209
Puredata PDP	v1z, fefo, hick209
ChucK	rfabbri, automata
ChucK MiniAudicle	rfabbri, automata
WebRTC	automata
OSC-Web	automata
Web-PD-GUI	automata
Live-Processing	automata
ChucK-Wiimote	automata
Audiolet	automata
Extempore	automata

group focused on a specific theme like video, audio and web. They were financed by contracts and support of the "Pontão de Cultura Digital: Nós Digitais". In Table 2 we can see a list of the free software created by "Lab Macambira" since July of 2011.

As of this writing the "Lab Macambira" have many software developers, and some of the trained developers continue to work voluntarily in the project.

5 Conclusions

In a scenario where GSD is growing as a popular form of software development, not just on free software projects but in the whole software industry, there is an increased need for methodologies to deal with it's disadvantages and at the same time to amplify it's advantages.

This paper has presented a simple methodology to GSD, being the development conducted on large or small groups of software developers, working on different countries or even at the same room. The AA methodology implements a simple system where each developer takes notes of his work, generating a periodical log of small text sentences. The sum of those sentences, along an entire session of work, results in a complete report. The report is made public available through a Website and are validated by other developers sorted aleatory by the AA Web server.

Instead of a merely work-management tool, AA acts as a methodology to improve the time sense of individuals, dividing their work on small sessions, and also reducing the need of extensive reports or unnecessary meetings. By asking users to write a minimal text sentence as a continuous log, AA minimizes disturbances: developers just type some characters, hit *enter* and go back to coding, which can help mental organization about the work being done.

AA application is not restricted to software development. As of this writing there is a comic book studio[5] using AA to manage their activities. People with non-software background, like social scientists, musicians and activists has also been using AA and contributing for its improvement.

For software developer teams, the use of AA was used successfully to auto-regulate the work of "Lab Macambira", a group of free software developers from Brazil. Since July of 2011 the group have contributed and created new free and open source software for a vast number of applications.

There are many aspects of the work which remain unfinished. New ways to report logs — the "Twitter like" messages, sometimes called "shouts" — from different interfaces like IRC, Internet Messaging services and email can make the use of AA easy and widespread, turning AA an ubiquitous system, presented on everyday communication channels. Even the work logs generated since July of 2011 could be statistically analyzed to recognize patterns in the behavior of individuals and their productions.

Being a free software system and an open methodology, AA could be used to auto-manage groups working on software or other kinds of activities. In this

[5]Pula Pirata Comics: http://pulapirata.com

Table 2: Software projects, created by "Lab Macambira", with the use of AA, since July of 2011 with a short description and the technologies — like programming languages or frameworks — involved. It is interesting to note the heterogeneity of projects and its areas of application.

Application	Description	Technologies involved
AA	Algorithmic Auto-regulation	Python, PHP
Ágora Communs	System for on-line deliberations	PHP
SIP	Scilab Image Processing toolbox	C, Scilab
animal	An Imaging Library	C
TeDi	Test Framework for Distance Transform Algorithms	C, Shellscript, Scilab
Macambot	Multi-use IRC Bot	Python
"Conferência Permanente"	Platform for the permanent conference of the rights of minors	PHP, JavaScript
CPC	Center for accounting of the Brazilian culture representation groups	Python, Django
Timeline	Interactive time lines on the Web	JavaScript
Imagemap	Interactive marking for on-line photos	JavaScript
ABT	Program for real-time execution and musical rhythmic analysis	Python
EKP	Emotional Kernel Panic	Python, ChucK
SOS	Aggregation and diffusion of popular and native knowledge about health	Python, Django
Creative Economy	Platform for creative, collaborative and solidarity economy of the culture hubs and cultural entities	Python, Django
OpenID Integration	Adaptations to existing software for unified login through OpenID	PHP
pAAinel	Dashboard for the real-time visualization of Lab Macambira activity	Python, Django
Georef	Collection of scripts to be used as reference, which aims to be a GIS platform to map public data of use to citizens	Python, Django
AirHackTable	Software for an instrument which generates sound from flying origami tracked by webcams	Puredata, C/C++, Scilab

way, there is interest on spreading AA to have even more developers contributing in a collaborative way.

Acknowledgments

Authors thank AA: the present research and even this manuscript was written using AA. The complete log is on-line at http://www.pulapirata.com/skills/aa. Authors are also grateful to IFSC/USP, IPRJ/UERJ, IFT/UNESP and all AA users and collaborators, spacially those who coded AA hacks for shell and bot uses and those who coded the different Web interfaces in current use.

References

[1] Global software development and delivery: Trends and challenges. 1

[2] Global software development: Who does it? 1

[3] Battin, R., Crocker, R., Kreidler, J., and Subramanian, K. Leveraging resources in global software development. *Software, IEEE 18*, 2 (2001), 70–77. 3

[4] Carmel, E. *Global software teams: collaborating across borders and time zones*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. 2, 3

[5] Carmel, E., and Agarwal, R. Tactical approaches for alleviating distance in global software development. *Software, IEEE 18*, 2 (2001), 22–29. 3

[6] German, D. M. The gnome project: a case study of open source, global software development. *Software Process: Improvement and Practice 8*, 4 (2003), 201–215. 1, 2, 3

[7] Komi-Sirviö, S., and Tihinen, M. Lessons learned by participants of distributed software development. *Knowledge and Process Management 12*, 2 (2005), 108–122. 2

[8] Last, M. Understanding the group development process in global software teams. In *Frontiers in Education, 2003. FIE 2003 33rd Annual* (2003), vol. 3, IEEE, pp. S1F–20. 1