Process Management Software as a Script, and the Script as a Pattern

Víctor Hugo Medina García, Sandro Bolaños Castro, and Rubén González Crespo

***Abstract*—This article presents the concept of script as the linchpin in managing the software process, the pattern is proposed as a script within the software process, is presented as a research result of the comparison of two software projects, which the radical difference is found in the script.**

All of these patterns, including traditional processes (Waterfall, Spiral, Big-bang, Prototype, etc.) and Agilists methodologies (XP, Scrum, FDD, ASD, Evo, and others), can be used as tool scripts using Colossus in which one component is supplied for this purpose. The aim of the component patterns of the process is to provide a tool to organize and customize the scripts, the script component coined the term as equivalent to the script. Finally it presents a software component that compiles a set of process patterns that can be used and generated automatically.

***Index Terms*—Software process, script, pattern, component, tools.**

I. Introduction

A script is defined by the Royal Academy of the Spanish language in the dictionary "El Pequeño Larousse", such as "*text that contains all the development of a film, radio, or television with all the details*."

The software development processes also have a script made up of activities and roles that the execution, as in the conventional script, the software process defines the details for their implementation, as well as the work of the director of a film or a play, the software project manager for the project tends to be managed properly to have a guide that provides a good degree of security that will form in the process of software project.

The script provides process management software itself but the synergy [1] of the roles is that the guarantee or hinder its implementation. In the software process there is a wide range of scripts but the same script is never attended the same way, the process scripts can carry out the project, but can also be a failed project, as in the movies that are reproduced, they either have been successful or a terrible failure, this leads to the premise that "the software developments process script is as important as one who embodies and who leads it."

Manuscript received February 14, 2012; revised April 20, 2012.

Víctor Hugo Medina García and Sandro Bolaños Castro are with Universidad Distrital Francisco José de Caldas, Bogotá, Colombia (e-mail: vmedina@udistrital.edu.co, sbolanos@udistrital.edu.co).

Rubén González Crespo is with Universidad Pontificia de Salamanca Madrid, Spain (Tel.: + 57 3114546209; fax: + 5713239300 ext. 1402; e-mail: rubenagc@gmail.com).

II. The Management as the Foundation of the Script

A script is a dead letter until it is embodied and staged, that it meets the management role, managing a script means to carry out the recommendations, follow the flow of work and control the variables that affect it. Of course if there is no management there is no script, but equally it can't be done if there is no script or management, they are inseparable from each other. Management leads to the realization of the script in hand of roles performed by people who need to engage in the process.

As in a play or a film, the action of seeking harmony roles leads to the success of the project [2]. Management should be focused primarily on the resources for the people who execute roles and the time needed to carry out the activities of the process, these two factors are critical to avoid delays in the projects [2]. Management process should be in hands of a leader of the process that will do so as a film director, to keep the synergy between roles "actors of the process." Management is the guideline for conducting a controlled process. In essence if you think for example in two scenarios in which you have used the same process, but one of them has failed while the other is successful, the question arises: What is the difference between the success of one and failure of the other? Logically, there can be many causes but the one we want to analyze with the discussion of the script is the way in which roles are embodied in this booklet, carried out by management.

A. Advantages and Disadvantages of the Script

The script is a software process that allows you to visualize the activities to be undertaken, also allowing the two key resources to plan a project: people in their various roles and time. What follows are two scenarios, the first point to a chaotic process "Big Bang" [3], the second scenario proposes a process with a well-defined script.

Apparently, it could be argued that a process can do without, but in essence when performing an activity, it is already a process. In the extreme case of lack of activity is a empty process by encouraging the principle of non-absence of the process [4], this extreme case is the least typical, rather it is usually not clearly identified activities, which it does not mean the absense of it, this case can be called a chaotic process, it is curious that garage projects [2] come to fruition when they are not supposed to use a software development process formally. In this scenario there is a high introspection, inner drive is used to solve a problem [5], of course the exception does not make the rule, these approaches are no longer quixotic efforts of battles against windmills like giant processes seen but are not present.

The second scenario proposes to follow predefined activities and configured in a wide range of processes and methodologies fairly widespread. Regardless of the proposal received, the common factor is that they are endowed with activities, tasks, phases, practices, etc.., Illuminating the path to be followed to reach that much desired success. The processes have been identified as classically conventional rigorous large volumes of documents, formats, standards are proposed pachyderms, which have emerged of complex and have enacted the holy grail of the processes. However, hard loads are placed on the projects and end up being the sword of Damocles for the same project.

It is clear that in these circumstances, rather than the script being beneficial it causes trauma. On the other side of the scale such as wind-driven steeds agile methodologies are based heavily on pragmatic practices that lead to obtaining viable and visible results in relatively short times, the script in this scenario is the speed set point and breaks rigorous documentation schemes, use of sophisticated tools, contracts and fixed plans, to make way for fast delivery, value people, affective communication and acceptance and change management, these approaches generally proposed for small and medium projects.

The intellectual model that you have is that conventional methodologies address to both large projects such as light address small projects and medium projects that it is like comparing a marathon runner vs a hundred meter dash runner. It is clear that they are hardly converging. If you have a script you have a play if it does there will be only one intention, one can ŕ ignore these processes or scripts can do without.

III. Research Results

TABLE 1: Comparative Project

Artifacts	Project A	Project B
Requirements	- Text Descripci ón	- Uses of Cases - Documentation of Philosophy - Glossary
Architecture Design		- View information structure - Roadmap View - Classes - Sequence - Activities - Systems - States - Components
Implementation	- V1.0 Editor - V2.0 Debugger - V3.0 Tester - V4.0 Semantic Editor - V5.0 Colossus Textual	- V1.0 Class Diagram

Below is the development of two software projects, the script used to differentiate, the projects were adjusted to the following conditions:

- The development time was one year.
- The development platform was Java in eclipse environment.
- To ensure conceptual integrity [2] the number of participants is the same n = 1.
- The projects vary in nature to avoid the effects of the second system [2].
- None of the projects had an antecedent.

The artifacts produced by the project are listed in Table 1.

For the project A to the end of a year of development was achieved by an executable with the following characteristics:

- Delays occur in the administration of complex files.
- There is a real-time recognition of inconsistencies in the files.
- The inclusions are not automatic.
- Only available to java.

This project carried out the following phases: requirements and implementation see Fig. 1. In the requirements phase, carried out in three months, was established in textual, the most used features robust editors like Eclipse and Netbean established only a few. In the implementation phase, conducted nine months later, we obtained five versions that evolved from the publisher through the debugger, tester, semantic editor to finish in the latest version of integration

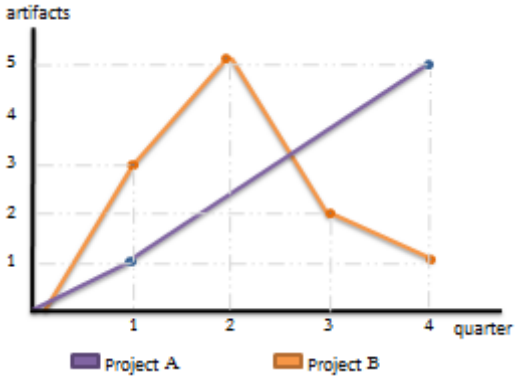

Fig. 1. Statistics projects A and B. Source: Authors

For Project B at the end of a year of development was achieved by a beta version with the following characteristics:

- They got only the class diagram in accordance with standard UML 2.X
- It is friendly in its use because of its simplicity and consistency.
- It performs the code generation.
- Response times are good as well as printing and image export format.
- No interoperability with other editors.

For this project there were four phases: requirements, designs, architecture and implementation see Fig 1. In the requirements phase, conducted in the first three months, were obtained 3 artifacts, use cases, a document of environmental philosophy and a glossary of the terms. In the design phase, then completed in the next three months, were obtained diagrams: class, sequence, activities, systems, and components states. In the architecture phase, held later in the next three months, we obtained the views [6]: Structure of information and route map. In the implementation phase later on in the last three months of the project, we obtained a beta version of class diagram.

According to the two projects, there are the following considerations: to recover project A , it has created

complications in the latest version and has been sought only to stabilize it.

Project B has continued and after another 7 months of work are total UMLV.2.0 diagrams [7], [8], there is an established architecture and modifications have been done without injury thanks to the existence of this architecture. As a result of software structure was obtained by developing a graphic framework that underlies part of the UML framework and new projects that need graphs.

Among some thoughts on the two projects are:

1) Project B used a script while in A was chaotic and focused on implementation.
2) Project B identified valuable artifacts to continue it, while Project A produced only executable versions difficult to continue to have no supporting documents.

The two projects despite having similar conditions differed in the development process. For the specific case of project B, metaprocess [9], [4] were used of software development in its capacity as management. This development process in these conditions set the selection as a key element within a wide range of processes, the most suitable process model according to the project and its environment. If you do not have a fit, proposes putting together its next phase of the process architecture is needed. For Project B, the process adjusted was RUP [10]. For project A, the process was chaotic. Ultimately, the script gives that gives us a process is fundamental for the development and evolution of software.

IV. The Script as a Pattern

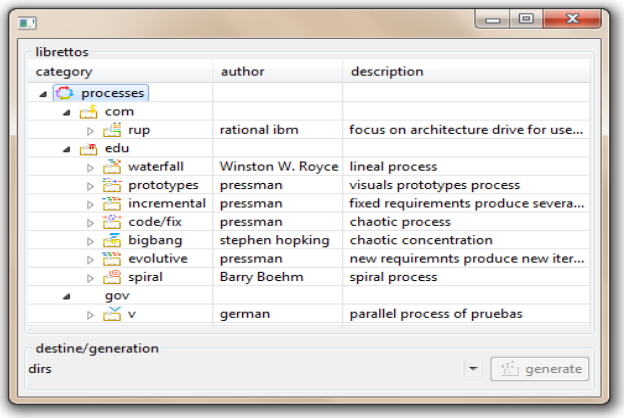

Fig. 2. Process Patterns. Colossus Software www.colosoft.com.co. Source: Bola ños

The script as the chart of the process is a mechanism that is reused again and again, to the recurring problem of having to have a guide for conducting a software development project. In this order of ideas, the script is a pattern, this concept has been popularized in the software with the work of Gamma [11] in the Gang of Four, the work of Shaw and Garlan [12], in the case of a pattern, it suggests different ways of organizing the activities which will manage the project. How to structure and compile recommendations on the process defines different patterns. There are two main aspects of the proposed process patterns in conventional processes and agile methodologies proposed. Among the most important are conventional processes [13]: cascade model, spiral model, model coding, repair, big-bang model, prototype model among others. In agile methodologies are: XP, Scrum, FDD, ASD, EVO and others, some governmental efforts such as V3 Metrics are added to the spectrum of process patterns. All these patterns can be used as scripts using the towering tool which provides a component for that purpose. The objective component of the process patterns as shown in the graph Fig. 2, is to provide a tool to organize and customize the script - the script component coined the term as equivalent to script.

The pattern features are defined through the script component which establishes the source of creation and the activities offered by the employer. The source of creation can be mainly: academic "edu" commercial "com" government "gov" and organizational "org". Patterns can also be organized methodologies see Fig. 3, they can even be mixed.

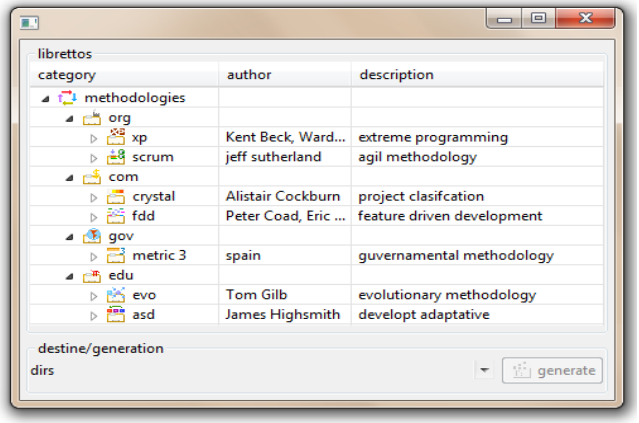

Fig. 3. Patterns of Methodology. Colossus Software www.colosoft.com.co. Source: Bola ños

As these activities are arranged through packages that serve as the repository of artifacts that are produced in each activity, although the equivalence between activity and package is not the best form of traceability as an activity represents an action as package represents an entity expressed preferably through a name, this equivalence may be an acceptable approximation and especially valuable for purposes of organization and control of a project. The component automatically generates the script that you want to follow, this organization is a hierarchy of activities and packages see Fig. 4.

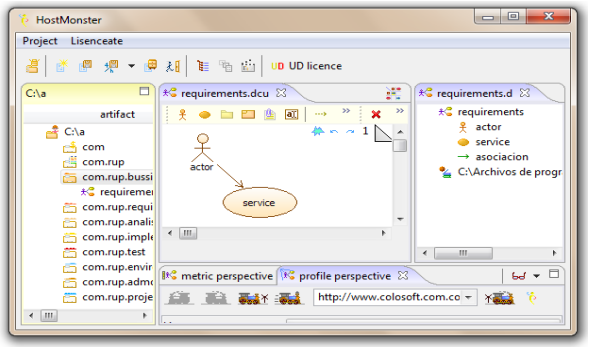

Fig. 4. Representation of patterns through packages. Colossus Software www.colosoft.com.co.

This representation can also be associated as a pattern from a processing graph, expressed through the process modeling language SMPL for its acronym in English "software process modeling language" proposed by the author Sandro Bola ños.

The SPML language proposes a set of basic building blocks to express a software process and connects directly with the component scripts, enabling more complex concepts introduced Fig.5.

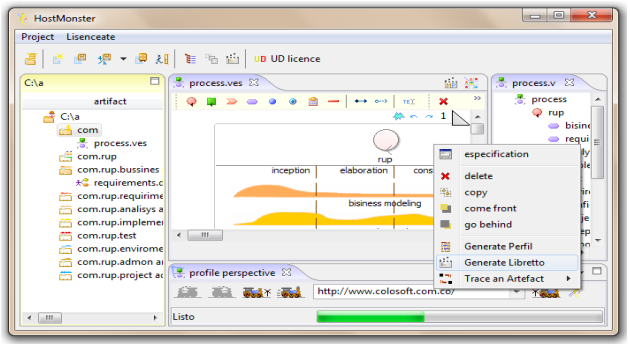

Fig. 5. Generation of patterns from SPML. Colossus Software www.colosoft.com.co. Source: Bolaños.

V. Conclusions

Software processes rely on scripts that become patterns for its repeated use, however the fundamental difference from one project to another rests on the roles that embody the script and of course on who runs it.

The specialty of the script can produce a greater chance at achieving the ultimate goal, it is much more productive to follow a script well established than a chaotic script.

Scripts can be treated as containers, through proposed traceability [14], from the activities to the packages; finally they organize the artifacts that are produced in the development process.

It is possible to have a better control of a software process, if this is managed through concepts like the script. In this order of ideas it can be seen as a script, the script, a series of activities to be performed by roles and finally some of the activities available as a pattern.

References

[1] J. O Connor, and I. McDermott, "Introduction to Sistemyc Thinking, Essencial Resources for Creativity and Resolution of Problems," Urano Editions. 1998.

[2] F. Brooks, Jr, *The Mythical Man-Month, Essays on Software Engineering*, Addison-Wesley, 1995.

[3] R. Patton, *Software Testing*, Second Edition, Sams Publishing, 2006.

[4] S. Bolaños, V. Medina, and J. Soto, "Metaprocess of Software Development. V International Symposium on Information Systems and Software Engineering in the knowledge Society," District University Francisco Jose de Caldas. Bogot áColombia. 30 of september of 2010a. www.udistrital.edu.co

[5] K. Popper, "Realism and the object of Science," l Tecnos Edt.. Pag. 45, 1998.

[6] Open Group, "Technical Standar, ArchiMate 1.0 Specification," Published by Open Group, february 2009. http://www.opengroup.org/

[7] G. Booch, J. Rumbaugh, and J. Ivar, *The Unified Modeling Langauge Reference Manual*, Second edition, Addison-Wesley, May 2005.

[8] G. Booch, J. Rumbaugh, and J. Ivar, *The Unified Modeling Langauge User Guide*, Second edition, Addison-Wesley, February 2006.

[9] S. Bolaños and V. Medina, "Software Development Process based on Kanowledge Management," Knowledge Management in Organization International Conference. University of Pannonia. Veszprem Hungri. 18 y 19 de mayo de 2010b.

[10] G. Booch, J. Rumbaugh, and J. Ivar, *Unfied Process of Software Development*, Addison-Wesley, February 2000.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[12] M. Shaw and D. Garlan, *Software Achitecture, Perspectives on an Emerging Discipline*, Prentive Hall, 1996

[13] R. Pressman, *Ingenieria de Software, Un Enfoque Práctico*, Mc Graw Hill. Sexta Edicion, 2008.

[14] S. Bolaños, V. Medina, and L. Joyanes, "Formalization principles for software engineering," *Engineering Magazine. District University Francisco Jose de Caldas*, vol 14, no. 1, pp. 31-37, 2009.