DOI: 10.25768/fal.ec.n29.a04

On software and television: Analyzing a minimal process for software requirements specification by TV broadcasters

Carlos Eduardo Marquioni
Programa de Mestrado e Doutorado em Comunicação e Linguagens, Universidade Tuiuti do Paraná
E-mail: cemarquioni@uol.com.br

Abstract

In this paper it is addressed the theme of software applications (*apps*) development and supply by TV broadcasters in order to synchronize the ads of TV shows' sponsors between multiple screens. That sync is required because audience is using multiple gadgets to watch TV (for example, during the materialization of viewers' second screen experience). Although broadcasters' supply of *apps* to the audience constitutes an alternative to the ads sync, such supply requires a previous software development. The execution of a minimal set of activities to software development (particularly to software requirements specification) is analyzed in this paper, observing that such activities should minimize impacts in the typical production processes of TV to mitigate the risk of broadcasters losing their focus on TV content production while developing the *apps*. The presentation of (i) Software Engineering processes' tailoring and of (ii) agile methods contributes to analyze a minimal process for software requirements specification.

Keywords: TV production; requirements engineering; television software studies; software engineering; processes' tailoring; agile methods.

Introduction

In this paper it is presented what seems to constitute an extension of Production Studies regarding TV processes. Such extension, provisionally entitled by the author of this paper as *television software studies*, is under development in an ongoing research project that applies an interdisciplinary framework encompassing TV Production Studies, Software Engineering (requirements engineering) and Project Management. Particularly this article addresses an outline of *television software studies* regarding the software requirements specification by TV broadcasters. Such specification is necessary to broadcasters be able to supply the audience with applications (from here *apps*) that sync the ads of TV shows' sponsors between multiple screens. Indeed, the sync seems

Data de submissão: 2018-08-21 . Data de aprovação: 2018-11-17.

A *Revista Estudos em Comunicação* é financiada por Fundos FEDER através do Programa Operacional Factores de Competitividade – COMPETE e por Fundos Nacionais através da FCT – Fundação para a Ciência e a Tecnologia no âmbito do projeto *Comunicação, Filosofia e Humanidades (LabCom.IFP) UID/CCI/00661/2013.*

applicable in the case of second screen experience (presented later in this article), as well as in the case of software development to support other forms of content distribution (like in the *TV everywhere* platform, that also needs the usage of an *app*).

However, to supply an *app*, it is necessary to develop it. And such development, when it is considered the process of TV show production, has associated the risk of deviate the broadcasters' focus: from TV content production to software development. Analysis and reflections related to an alternative to mitigate such risk are presented in this paper, considering the definition of a minimal requirements specification process that encompasses notions of Software Engineering processes' tailoring and agile methods. Before presenting the alternative itself, there is the need of explanations related to (i) the general context analyzed in this paper, and to the associated (ii) need of ads sync.

The (i) general context of this paper considers the second screen experience in Brazilian terrestrial commercial television, as well as the relation between the audience, broadcasters and advertisers (social actors of televisual ecosystem). Regarding second screen, it is worth noting that the associated experience encompasses a complex scenario related to cultural aspects. The complexity of the scenario can be observed when considering culture as a whole way of life that has its meanings redefined (in an *in process* redefinition) (Williams, 1989: 8) by the social actors when these actors have contact with (and use) cultural materialities all over the years – or *in the duration*. The understanding of the scenario involves also observing that to viewers, watching TV according to the broadcasters' defined TV shows schedule continues to be an effective practice in some countries. This is the case in Brazilian terrestrial commercial TV.

As a preliminary information related to terrestrial commercial television in Brazil, it can be pointed that Brazilian audience watches TV in "flow" (Williams, 2005: 89-90) during about "four and a half hours a day", as Carlos Safini (senior executive of *Globo Play's TV Everywhere* service in 2016) stated during his panel at CIAB Febraban in 2016. Thus, televisual media maintains its cultural relevance, what can be noticed with the audience ratings that terrestrial commercial TV channels reach in Brazil. Additionally to the high consumption of "linear" TV (Douglas, 2015) in the country, it is also possible to identify the existence of *in progress* cultural reconfigurations that cover some obvious changes in TV ecosystem (like the images digitalization or the increase of devices where viewers can watch TV). Associated with these obvious changes, there are more sophisticated ones. In this paper the scenarios classified as more sophisticated are those that tend to be *almost invisible*, mainly because they tend to be related to cultural redefinitions.

In the case of Brazil, data released by the "Ibope Inteligência [Institute informed that] Brazilian audience is becoming a multi-screen one: 88% of Internet users *watch TV while accessing the Internet* by smartphone (65%), computer (28%) or tablet (8%)" (Lopes and Greco, 2016: 139; my emphasis). While it is possible to notice that more people are watching TV concomitantly to Internet access (using a mobile device), the sophistication and *invisibility* previously mentioned are related to the fact that the public executes such actions *almost without thinking about it*. It occurs potentially due to a cultural reconfiguration related to the way of watching TV promoted by the multi-screen aspect, which has been referenced using mainly two terms:

> The term (i) social TV is adopted when the usage of the gadget connected to the Internet covers audience's posting of comments related to the aired TV show. These comments are posted in digital social networks (like *Facebook or Twitter*) [. . .]. The (ii) second screen (or even multiple screens) term is used to name the process executed by audience to search on the Internet for additional information related to the content aired on TV (Marquioni, 2016: 1-2).

In this paper, the expression *second screen experience* is used to encompass both denominations (social TV and second screen). Additionally, it is considered that the reconfiguration in *TV experience* by viewers has high relevance in the domain of broadcasters. Thus, despite such experience is materialized by the viewers (by the audience), it potentially impacts the TV shows production (by broadcasters) and the TV shows sponsoring (by advertisers).

To understand the relation encompassing these three social actors in televisual ecosystem in the case of second screen experience and (ii) the associated need of ads sync between gadgets, it can be pointed that when using a technological device connected to the Internet (and accessing a search tool and/or a digital social network) viewers are able to materialize their second screen experience independently from broadcasters. Such *independence* can eventually show to the public (in the second screen dispositive) contents of competing broadcasters (in relation to that they watch on TV, on the first screen), or even it can be shown to the audience ads of competing advertisers (in relation to the sponsors of the TV show aired to the first screen). This scenario can be analyzed as a complexification of zapping navigation between TV channels, because viewers can create with the second screen experience a mosaic between devices screens (and not between TV channels, as it occurred in the case of *traditional zapping* using the remote) while they materialize the *independent* experience. And such mosaic can cause a kind of "distraction" (Proulx and Shepatin, 2012: 106) due to attention sharing between the multiple-screens, potentially impacting the sponsorship of TV content production if advertisers consider the *distraction* inappropriate due to the risk of reduction in ads reach.

As an alternative to broadcasters have *some control* of the scenario, and even minimize the effects of the distraction with the use of multiple screens, it can be pointed the supply of *apps* to the audience. Such supply could enable, for example, ads synchronization between screens (Carneiro, 2012: 152).

But the *app* supply by broadcasters (which presupposes software development) potentially promotes variations in the typical TV production process. And it should be observed that even when the *apps*' development is executed by subcontractors there is a need of *integration and synchronization* between software development and TV production.

To address such complex scenario, in this article Software Engineering processes are presented in the next two sections – particularly regarding the outline of *apps* technical specification. In *The tailoring of traditional processes and agile methods: a conceptual analysis*, it is pointed that processes' tailoring constitutes an alternative that proportionate agility to development of *apps* even when using traditional Software Engineering approaches. The section *On a minimal, tailored and agile software process specification* it is presented a selection of artifacts that seems to constitute an alternative (in both practical and conceptual terms) to software development by broadcasters, encompassing agility and formalization. A minimal set of technical artifacts and processes is considered, allowing to broadcasters to keep their business focus in producing TV shows, as well as enable control to the *apps* development (even when subcontracting the software development).

The tailoring of traditional processes and agile methods: a conceptual analysis

It is possible to notice two main perspectives in Software Engineering bibliography. One of them is related to (i) traditional development processes, and encompasses specification, development, tests and management; the other one regards (ii) agile development that, as its name suggests, would propitiate agility to software development (more specifically, it would be a more agile process than the traditional development). This agility would be reached mainly due to the almost elimination of software specification when using the agile perspective. The main argument presented to justify the (ii) agile approach is that the specification in (i) traditional processes cause an excessive bureaucracy that make it difficult to promote changes in requirements. Undoubtedly agility must be considered relevant especially in the case of software development by broadcasters – since broadcasters' business is TV content production (and not necessarily software development). However, some specification is required because it is necessary to integrate and synchronize different life cycles in order to produce different products that must be launched and must operate concomitantly (the TV show and its *app*). Additionally, the requirements specification can be considered as a critical factor when subcontracting software development (what has become usual in Brazilian development related to *TV Everywhere apps*).

Thus, in this section, the perspectives (i) and (ii) are tensioned considering the hypothesis that an ideal scenario should enable both agility and specification using a *minimal requirements specification process*. The analysis of this paper allow inferring that processes' tailoring can proportionate agile development particularly because

> Agile processes have emerged in certain development communities and projects as a reaction against overly heavyweight practices [of Software Engineering], sometimes resulting from the misinterpretation of process models and the amount of 'ceremony' and reporting they require (Van Lamsweerde, 2009: 54).

It is presented below a brief theoretical analysis encompassing the (i) traditional processes and the (ii) agile methods in order to make reflections on the viability of having agile development even with the use of traditional processes (with software specification).

To start, it is worth mentioning a statement related to Scrum's agile method, according to which it "is nearly impossible to develop software in short periods of time [with agility], with high quality and with a low budget using the 'defined and repeatable' process approach [of traditional perspective]" (Schwaber and Beedle, 2002: 110). The used terms *defined* and *repeatable* refer directly the jargon of the software quality model entitled CMM (*Capability Maturity Model*), which had its acronym updated in early 2000's to CMMI (*Capability Maturity Model Integration*) (Chrissis; Konrad and Shrum, 2010) and is obviously associated in the quote with the traditional perspective of software processes. Scrum authors justify their statement informing that "processes to manufacturing" (Schwaber and Beedle, 2002: 106) would not be applicable in the context of software development. Complementally, the authors point that it would be illusory the possibility of stable requirements in a software project: "Requirements never stop changing. It makes little sense to pretend that this is not the case and attempt to set requirements in stone before beginning design and construction" (Schwaber and Beedle, 2002: 34). The Agile Modeling perspective also states that it would not be possible to "'freeze' the requirements [. . .] [, because] changes [in both business environment and software requirements will] happen" (Ambler, 2004: 98). The practical result associated to such statements was that some software professionals reduced the software specification, even reaching "cycles of dueling methodologies for software 'engineering,' without a true foundational theory to unite them. In the end, many of these methods did not even address the true needs of the skilled craft practitioners of the industry" (Jacobson and Seidewitz, 2014: 51).

Regarding the affirmation that requirements will change, a brief bibliographical revision allows noticing that such perception was also addressed in the year 1979 (in relation to traditional specification methods): "the freezing specification is a myth" (Demarco, 1989: 280). Later, in early 1990's, Requirements Engineering authors repeated the assumption: "Requirements changes occur while the requirements are being elicited, analysed and validated and after the system has gone into service. Requirements change is unavoidable" (Kotonya and Sommerville, 1998: 115). The authors of traditional processes also informed that it would be expected so many changes that stable requirements should be defined as the ones that "*change more slowly* than volatile requirements" (Kotonya and Sommerville, 1998: 116, my emphasis) – but even stable requirements change.

Once both perspectives, the (i) traditional development and the (ii) agile one apparently say the same thing in relation to changes in requirements, a reflection that can be pointed is that the van Lamsweerde statement quoted previously (regarding the misinterpretation in relation to traditional process models) tends to be correct. Indeed, that misinterpretation seems potentially related to a lack in bibliographic revision during agile methods definition. Because both approaches agree with the theme of requirements, it seems possible to unify them. But the requirements change is not the one and only conceptual *coincidence* between the perspectives, as presented in the next paragraphs.

Regarding the bureaucracy pointed by the agile perspective in relation to the (i) traditional processes, the execution of a bibliographical revision allows observing frequent statements related to the need of process tailoring to address the issue in software development projects. However, the obligation in the execution of activities and/or in creating all the artifacts described in the process seems applicable only when the standard process does not provide tailoring instructions as presented in managerial guides. Thus, it is fundamental to project managers the selection of "development methods for their projects" (PMI and IEEE Computer Society, 2013: 17). Such selection encompasses not only getting the methods and artifacts from the standard process, but also the analysis of the relevance in their usage in each case:

> In the context of software engineering, a process is *not* a rigid prescription for how to build computer software. Rather, it is an adaptable approach that enables the people doing the work

(the software team) to pick and choose the appropriate set of work actions and tasks (Pressman and Maxim, 2015: 16, italics in the original).

In an Object Oriented development, for example, the process tailoring execution must define which artifacts (from the standard set) need to be created, even considering that the Pareto Principle is applicable to UML's set of diagrams: "You can model 80 percent of most problems by using about 20 percent of the UML" (Rosenberg and Scott, 2001: 1).

To guide the process tailoring, it is possible to highlight the CMMI software quality model previously mentioned; More specifically the IPM (*Integrated Project Management*) process area defined in that model informs that its purpose is "to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined *process* that is *tailored from the organization's set of standard processes*" (Chrissis; Konrad and Shrum, 2010: 187, my emphasis). Thereby, "variability among projects is typically reduced and projects can easily share process assets, data, and lessons learned" (Chrissis; Konrad and Shrum, 2010: 188). Thus, process tailoring constitutes the alternative to make the process repeatable and defined; and agility as well, as discussed below.

In the early 2000's, the Unified Process also pointed the need of process tailoring. Particularly the Environment Discipline of Unified Process considers taking "the organization-wide process and further refine it for a given project. This level takes into consideration the size of the project, the reuse of company assets, the initial cycle [. . .] versus the evolution cycle, and so on" (Rational Unified Process, 2001)

To justify process tailoring, the bibliography related to *traditional* approaches indicates that the adopted/defined process to a project (in relation to both, a new development or a software maintenance in an available app) "should *be agile and adaptable* (to the problem, to the project, to the team, and to the organizational culture). Therefore, *a process adopted for one project might be significantly different than a process adopted for another project*" (Pressman and Maxim, 2015: 18-19; my emphasis), despite the projects use as reference the same organizational pattern processes.

Advancing with the tension between (i) traditional processes and (ii) agile methods, it is necessary to point that the agile perspective highlights the importance of adopting practices that involve the use of "iterative, incremental development" (Schwaber and Beedle, 2002: 4). The option in using a iterative and incremental approach was also presented previously by authors that defined traditional processes: in the year of 1988, the spiral model approach was presented as a development alternative that "provides the potential for rapid development of increasingly more complete versions of the software. [...] A spiral model is divided into a set of framework activities defined by the software engineering team" (Pressman and Maxim, 2015: 47-48). An update was presented in late 1990's by the authors of the Unified Process, when these authors indicated that to each software release (Jacobson; Booch and Rumbaugh, 1999: 85-107) it would be necessary to "plan a little", to "specify, design and implement a little", to "integrate, test and run each iteration a little" (Jacobson; Booch and Rumbaugh, 1999: 87).

Again, both the traditional and agile engineering perspectives seem to point similar statements, despite the use of different jargon.

Considering that an adequate usage of process tailoring tends to contribute to bureaucracy reduction, and to speed up the process of *apps* development, in this paper are adopted from here neutral terms to refer the software development context, observing that "an agile software process must adapt *incrementally*" (Pressman and Maxim, 2015: 70; italics in the original). Thus, the expressions agile development, agile modeling, agile methods, or traditional processes are avoided from this point on, especially not to suggest value judgment - for example, that the agile approach would characterize a better alternative in relation to traditional methods, or that traditional methods must be avoided. After all, "No one is against agility. The real question is: What is the best way to achieve it? [...] [And it is necessary to notice that] there is much that can be gained by considering the best of both schools and virtually nothing to be gained by denigrating either approach" (Pressman and Maxim, 2015: 71).

In this scenario, it is considered the possibility of defining a minimal process to software with the "sufficient detail" (Ambler, 2004: 30). And that includes visual specification, emphasizing that the use of the term *sufficient* involves establishing traceability between the abstractions of requirements, aiming to enable reliable impact analyzes during software maintenances. Additionally, considering that changes are a premise (and noting that traceability among the minimal proposed abstractions potentially contributes to the execution of impact analyzes and to subcontracting), more than courage to embrace changes (Beck, 2004: 48-49), it is pointed that the software technical team that attends the broadcaster should have objective conditions of assessing and applying adjustments in the process (tailoring it).

In other words, it is feasible to relate the practical experience of agile methods with the more traditional aspects of Software Engineering, culminating with an approach that integrates the TV production life cycle with the software development life cycle, also enabling formalism and agility to *apps* development. Such approach would have the advantage of not disrupting the main purpose of TV broadcasters that is to produce TV shows. Some key elements that enable a *minimal specification process* and seem to use a perspective that encompasses both perspectives (processes tailoring and agile methods) are discussed in the next section.

On a minimal, tailored and agile software process specification

In this section it is analyzed what seems to constitute an alternative to a minimal set of processes and artifacts to *apps* specification by broadcasters. The main argument presented is that a minimal process potentially enables TV channels to keep their focus on TV production (with only the enough emphasis on the processes of software specification and development). It also would make it possible to subcontract software development.

Regarding the mentioned minimal specification, in order to mitigate the risk related to misunderstandings in requirements elicitation and validation, the alternative presented here considers starting the specification process with prototypes development. This option is justified because prototypes enable meaning generation from mental models: in practical life "We usually construct a mental model when we are required to make an inference or prediction in a particular situation" (Stone; Jarrett; Woodroffe and Minocha, 2005: 78). During the design of interfaces, mental models are useful to apply metaphors in computational environments: Especially when there are difficulties in defining requirements, the use of prototypes tends to contribute with *app's* requirements elicitation.

It is worth noting that although some software processes – such as the ICONIX (Rosenberg and Scott, 2001) – point the creation of prototypes as Graphical User Interfaces (GUI) in the initial stages of the requirements process, in this article it is considered that semiotically,

> Even when requirements are written [specified] using natural language [in textual format], it is possible to generate an *effect of prototype* to the non-technical reader; Thus, while reading texts in natural language, the reader could be *induced to think in interfaces*. For such *effect* to occur it is necessary to define a pattern to requirements writing, and a morphology must be maintained: a set of *special* words seems adequate to establish such kind of convention (Marquioni, 2008: 160; italics in the original).

It is necessary to highlight that this approach does not require that customers specify the requirements, as defined with some agile methods (Beck, 2004: 66). In fact, the software team should proceed with the specification, potentially helping customers to identify possibilities of interactivity in second screen *apps*. After the validation of textual prototypes, visual/graphic prototypes should be created. Once this *two-level prototyping* (considering both the *textual prototype* and the *graphical one* as *levels of prototyping*) gets developed and validated, a more technical specification of the product should be executed with the creation of a set of technical diagrams to enable communication among software professionals. With this approach, instead of validating the *software behavior* separately from the designed interface (for example executing a technical validation with business users using the use case scenarios when developing software according to the Object Oriented Paradigm) (Rosenberg and Scott, 2001: 38), the business user can validate the behavior of the *app* from the prototyped interface, potentially increasing the probability of a successful understanding of requirements during validation.

Complementing the creation of prototypes in this *two-level approach*, the technical diagrams and process proposed by ICONIX (Rosenberg and Scott, 2001) seem appropriate for technical *software behavior* specification due to, at least, three major factors: (i) the details required to software specification in ICONIX, (ii) the use of prototyping that is aligned with the approach presented previously in this section and (iii) the minimalist set of artifacts suggested; These three factors are discussed below.

(i) Conceptually, ICONIX

> sits somewhere in between the very large Rational Unified Process (RUP) [related to traditional software development process] and the very small eXtreme programming [related to the agile perspective] approach (XP). [...] [This *intermediary* position is reached using a] subset of the UML [artifacts that] focuses on the core set of notations that you'll need to do most of your modeling work (Rosenberg and Scott, 2001: 1).

Directly associated with the content previously discussed in this same section, ICONIX suggests the adoption of (ii) prototyping: from "simple line drawings of your screens" (Rosenberg and Scott, 2001: 9), to "some rapid prototyping of the proposed system" (Rosenberg and Scott, 2001: 13). Indeed, ICONIX argues that prototypes "help define the use cases [...] [, enabling a] 'proof of concept'" (Rosenberg and Scott, 2001: 54-55). Thus, after the validation of the prototype, "the text [the textual technical specification of the use case model] for a given use case should match up well with the associated GUI elements" (Rosenberg and Scott, 2001: 55). It is established the traceability between the specification abstractions, that relates the *apps* interfaces with their textual expected behavior. In such scenario, the *textual prototype* previously mentioned constitutes a step that is executed before the "simple line drawings" suggested by ICONIX.

It is worth noting that ICONIX indicates the use of UML as the language to technical specification. This suggestion is particularly interesting considering that the understanding of the represented content is potentially increased because UML is a specification language adopted worldwide by most software professionals since late 1990's. However, it is possible to infer that any technical notation that allows the modeling of dynamic and static contexts in relation to the software product could be used to the specification. Thus, a TV channel that have already a software process defined according to the Structured Analysis Paradigm – or to the Essential Analysis one – could adhere to the minimal process presented in this paper, tailoring the process to suit the techniques. This alternative seems relevant when considering the main business of broadcasters: to produce TV content.

The subset of UML suggested by ICONIX consists of only "four different kinds of UML diagrams [...]. Limiting your focus to this core subset of diagrams will make a significant impact [even] on your learning curve as you learn how to do modeling with UML" (Rosenberg and Scott, 2001: 8); Additionally, "the approach is *iterative* and *incrementa*l" (Rosenberg and Scott, 2001: 10). Thus, using ICONIX seems adherent to the perspective of software specification as discussed in the previous section.

ICONIX process also reinforces the need of establishing a "sharp focus on the traceability of requirements" (Rosenberg and Scott, 2001: 1). Staring from the textual prototypes, the decoding of the requirements into visual prototypes and their adaptation to use cases traceability gets evident. Then, with the elaboration of the robustness diagram to fill "the gap between requirements and detailed design" (Rosenberg and Scott, 2001: 5), traceability can be defined with relative transparency: the boundary objects mapped in the robustness diagrams of each use case are the interfaces on which "actors will be interacting" (Rosenberg and Scott, 2001: 38) (mapped previously as text and line draw). The *boundaries* are classes that *realize* technically the validated prototypes, that are, in practical terms, "windows, screens, dialogs and menus" (Rosenberg and Scott, 2001: 62).

As a last step in the minimal process specification, each one of the complexes use cases scenarios (and only these scenarios), should be associated with a "sequence diagram that shows us which object is responsible for which function in our code" (Rosenberg and Scott, 2001: 4). The choice in modeling only the complexes scenarios using sequence diagrams is justified because in the case of *apps* to materialize the second screen experience, the greatest difficulties tend to be observed in cases of integrations between software systems (for example, when integrating the *app* with legacy systems or with digital social networks features). Thus, for most cases this specification tends not to be necessary.

The ICONIX process indicates that *preferably the activities to review contents should occur with* technical staff and business users "in a room together" (Rosenberg and Scott, 2001: 53).

The suggestion is appropriate especially considering that to business users it tends to seem more feasible than the *client fulltime working with the team* premise of eXtreme [sic] Programming – not only because the business professional could continue to perform his/her activities in the original business work environment, but also because the project team could request meetings with the clients to discuss identified issues when necessary.

Figure 1 presents graphically the adaptation in ICONIX as discussed in this section, considering the minimal specification process (and its link with code writing).

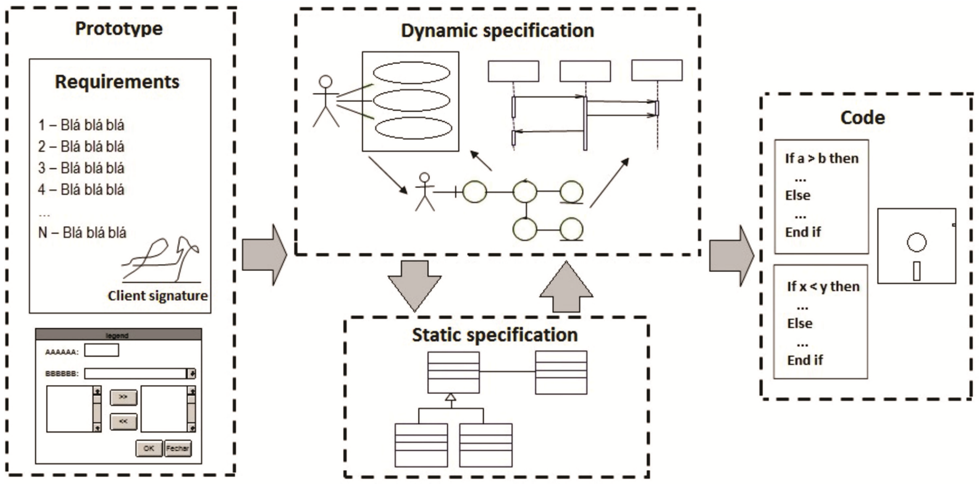

Fig. 1. Set of minimal artifacts for second screen *apps* development
Source: Adapted by the author from (Rosenberg and Scott, 2001: 09)

The approach (a) provides a minimal set of artifacts that guides the *app* development, and (b) keeps the business focus of terrestrial commercial broadcasters to TV content production. In addition, it (c) tends to contribute with the creation of a useful technical specification.

Final considerations

The ways of watching TV are been redefined with the use of multiple devices by the audience. In order to minimize the risks of advertisers reducing the sponsorship related to TV shows production due to the cultural reconfiguration in terrestrial commercial television ecosystem, it was pointed in this paper that broadcasters could enable alternatives to the materialization of the second screen experience by the public. An alternative is the supply of *apps* by the TV channels: such *apps* could sync ads between screens. However, the *app* supply tends to complexify the typical TV production process, due to the execution of activities related to software development life cycle during the TV production life cycle. To minimize impacts it could be defined a minimal software process (preferably considering the synchronization of TV production and software development life cycles). That minimal software process potentially mitigates the risk of broadcasters to deviate their focus from TV content production business. Instead of imposing the usage of a traditional method of specification or an agile one, this paper pointed that it would be preferable to apply process tailoring to the development of *apps*, leading agility to software development. In this sense, adaptations in ICONIX process constitute an interesting technical alternative. The project conducted by the author of this paper is on the run, investigating not only technical aspects (as those addressed in the present article), but also management alternatives to orchestrate the whole scenario.

The continuity of the research seems relevant because it is possible to infer that viewers can get an upgrade *status* in the *in progress* reconfiguration of the TV ecosystem context. Indeed, it seems possible to notice a kind of *migration* from an audience *tuned on* the TV channel, to an audience *connected to* the TV channel. This last one can enable an even more close relationship with the broadcasters, even contributing to the redefinition of sponsoring the TV production (for example, with the usage of *big data* resources). The minimal software requirements specification process related to the supply of *apps* by broadcasters seems to be fundamental to advance with such *migration* and *redefinition*.

References

Ambler, S. (2004). *Modelagem Ágil: práticas eficazes para a Programação eXtrema e o Processo Unificado*. Porto Alegre: Bookman.

Beck, K. (2004). *Programação eXtrema (XP) explicada: acolha as mudanças*. Porto Alegre: Bookman.

Carneiro, R. (2012). *Publicidade na TV digital: um mercado em transformação*. São Paulo: Aleph.

Chrissis, M.; Konrad, M. & Shrum, S. (2010). *CMMI for Development: guidelines for process integration and product improvement*. Boston: Addison-Wesley.

Demarco, T. (1989). *Análise estruturada e especificação de sistema*. Rio de Janeiro: Campus.

Douglas, P. (2015) *Future of television: your guide to creating TV in the new world*. Studio City: Michael Wiese Productions.

Jacobson, I.; Booch, G. & Rumbaugh, J. (1999). *The Unified Process Development Process: the complete guide to the Unified Process from the original designers*. New Jersey: Addison-Wesley.

Jacobson, I. & Seidewitz, E. (2014). A New Software Engineering. *Communications of the ACM (The Association for Computing Machinery)*, *57*(12): 49-54.

Kotonya, G. & Sommerville, I. (1998). *Requirements Engineering: processes and techniques*. West Sussex: John Wiley & Sons.

Lopes, M. & Greco C. (2016). Brasil: a 'TV transformada' na ficção televisiva brasileira. In M. Lopes & G. Orozco (orgs.), *(Re)Invenção de Gêneros e Formatos da Ficção Televisiva*. Porto Alegre: Sulina.

Marquioni, C. (2008). *Técnico vs. usuário: uma análise do processo comunicacional na Engenharia de Requisitos de Software*. Curitiba: UTP.

Marquioni, C. (2016). Sobre o desenvolvimento de aplicativos de segunda tela para a TV comercial: a sincronização de ciclos de vida e a emergência de uma audiência conectada (notas iniciais de pesquisa). *Proceedings of the 34th Congresso Brasileiro de Ciências da Comunicação – Intercom 2016*. São Paulo, 4th-7th September.

PMI & IEEE Computer Society. (2013). *Software Extension to the PMBOK Guide Fifth Edition*. Atlanta: Project Management Institute, Inc..

Pressman, R. & Maxim, B. (2015). *Software Engineering: a practitioner's approach*. New York: McGraw-Hill.

Proulx, M. & Shepatin, S. (2012). *Social TV: How Marketers Can Reach and Engage Audiences by Connecting Television to the Web, Social Media, and Mobile*. New Jersey: John Wiley & Sons.

Rational Unified Process. (2001). Rational Software, w/l.

Rosenberg, D. & Scott, K. (2001). *Applying use case driven object modeling with UML: an annotated e-commerce example*. New Jersey: Addison-Wesley.

Schwaber, K. & Beedle, M. (2002). *Agile software development with Scrum*. New Jersey: Prentice Hall.

Stone, D.; Jarrett, C.; Woodroffe, M. & Minocha, S. (2005). *User interface design and evaluation*. San Francisco: Morgan Kaufmann Publishers.

Van Lamsweerde, A. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Glasgow: John Wiley & Sons.

Williams, R. (1989). Culture is ordinary. In R. Gable (ed.), *Resources of Hope: Culture, Democracy, Socialism*. London: Verso.

Williams, R. (2005). *Television: Technology and Cultural Form*. Padstow: Routledge Classics.