Software Development Activities Catalogue Proposal

Danilo Martínez [a,b] , Xavier Ferre [a]

a. Universidad Politécnica de Madrid, Spain

b. Universidad de las Fuerzas Armadas ESPE, Ecuador

1 Introduction

Software development process is defined by [1] as "the process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use". The activities that a software process is composed of can vary according to the specific characteristics of the project and field of application.

There is not agreed upon software development process for mobile applications, with the project leader deciding upon his/her own criteria and experience.

We have created an integrated app development framework that includes a catalogue of candidate activities to be used for mobile app development.

2 Background

Mobile app development shares some similarities with traditional desktop software development. Nevertheless, the particularities of the mobile domain affect the software process and its activities. In order to structure our activity catalogue proposal, we have based on the results of two studies: First, we carried out a Systematic Mapping Study (SMS) on mobile software development process[1], and, second, we carried out a survey to 13 mobile developers [2]. Fig. 1 shows the activity map resulting from the SMS.

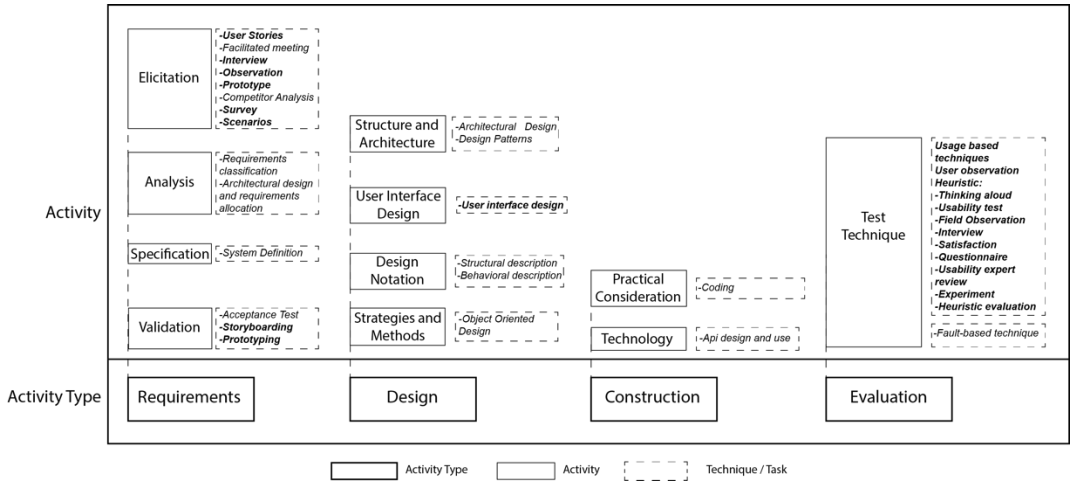

Fig. 1. Activity map resulting from the Systematic Mapping Study

In a similar manner, the activity map from the second study, the survey, is shown Fig. 2.

[1] http://www.grise.upm.es/sites/extras/20/pdf/SMSMobileSoftwareProcess.pdf

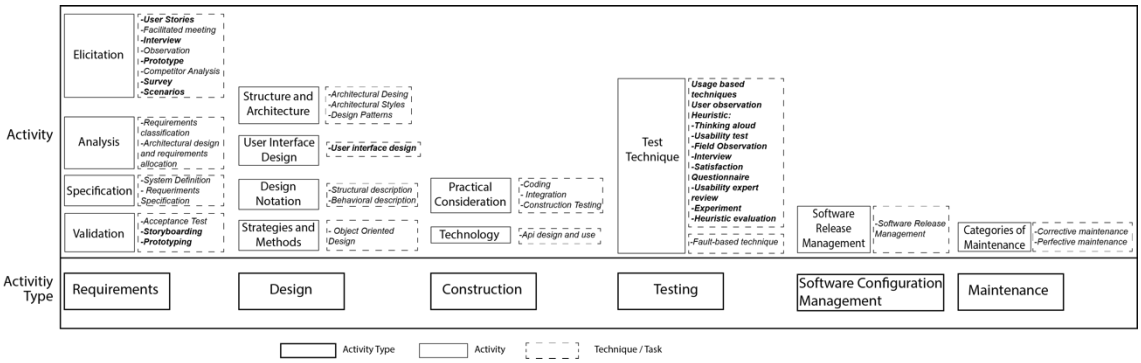

Fig. 2 Activity map resulting from the survey to mobile developers

3 Software Development Activities Catalogue

3.1 First Version

Having the two activity maps we proceeded to match them to obtain a fusion of the two. For this purpose, we compared each one of the activities, sub-activities, and techniques so as to avoid duplicities. The resulting combined map is shown in Fig. 3.

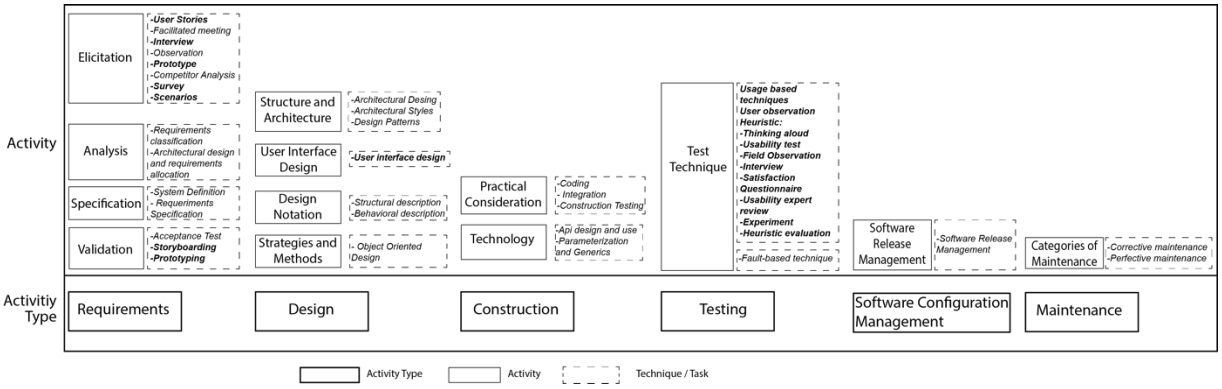

Fig. 3 Combined activity map – first version

Activities have been structured into six categories: Requirements, design, construction, testing, software configuration management, and maintenance. Please note that we are talking about activity types, not about process stages, since the latter would imply a waterfall life cycle approach, which is not the case. Mobile development project leaders will have the flexibility to choose the activities to be carried out in the moment they are needed, not necessarily in a sequential manner. Given that there is an important number of techniques from the Human-Computer Interaction (HCI) field, related to usability and User eXperience (UX), they have been highlighted in bold face in the figures. Usability and UX are acquire an special relevance in mobile applications due to the high expectations to this regards from mobile users, and to the fierce competition in the mobile app market.

This first version of the activity catalog is composed of 12 development activities and 39 techniques.

3.1.1 Evaluation

In order to evaluate this first version of the activity catalog we carried out two evaluation activities. In the first evaluation activity bachelor students from the *Universidad de las Fuerzas Armadas (ESPE),* in its campus in Sangolquí (Ecuador) used to catalog to choose techniques for the development of a mobile application in the summer of 2017. The second evaluation of the catalogue was carried out by 2 expert developers with an average of 3 years of experience in the development of mobile applications.

The results of these two evaluation efforts were as follows:

- The set of activities, sub-activities and techniques, as it is detailed in the catalogue conveys the idea of a list of prescriptive activities, that even may suggest a waterfall life cycle approach. This is not the aim of the catalogue, as it is offered as a guide with a set of activities and techniques from where the project leader can choose from.
- There is a set of activities and techniques which have received more interest from the bachelor students, suggesting that they could be of interest for novel developers. On the other hand, there is a high number of techniques not considered by any team. As a result of the expert evaluation we have identified that the activity catalog is too extensive. Reducing the number of proposed techniques, having thus a more compact catalogue would facilitate its comprehension, application and management. Agile approaches, very common in mobile development, favor simple solutions to complex ones.

3.2 Catalog Refinement: Second version

Considering the results of the evaluation in the previous section, we have defined the following criteria for keeping activities in the catalogue:

1. Techniques that have special relevance in the mobile field.
2. Techniques that appear at least twice in the SMS or once in the survey.
3. Techniques selected have been selected both by the case study with bachelor students and in the expert evaluation.

Techniques that do not comply with 1. and either 2. or 3. have been discarded, because they are considered as having low relevance. The results of this process are detailed in Table 1 to Table 6, where discarded techniques appear with grey background.

Table 1. Requirements Activities Refinement

Activity	**Task/Technique**	***Literature***	***Survey***	***Evaluation***	***Relevant in mobile***
Elicitation	User Stories	[3], [4], [5]		X	X
	Facilitated meeting	[6], [7], [8], [9]	[2]		X
	Interview	[7], [10]			X
	Prototype	[11]			X
	Competitor Analysis	[12], [11]			X
	Survey	[13], [14]		X	X
	Scenarios	[9]			X
	Observation	[7], [8]			
Analysis	Requirements Classification	[15]		X	X
	Architectural design and requirements allocation	[4]			X
Specification	Requirements Classification	[15]			
Validation	Prototyping	[3]	[2]		X
	Acceptance Test	[4]			
	Storyboarding	[9]			

Table 2 Design Activities Refinement

Activity	**Task/Technique**	***SMS***	***Survey***	***Evaluation***	***Relevant in mobile***
Software structure and architecture	Architectural Design	[15], [14],[4]		X	X
	Design Patterns	[8], [10],[11]			X
	Architectural Styles		[2]		
User interface design	User interface design	[15], [6], [4], [5]	[2]	X	X
Software design notation	Structural description	[3], [4]			
	Behavioral description	[4]			
Software design strategies and methods	Object Oriented Design	[15]			

Table 3 Construction Activities Refinement

Activity	**Task/Technique**	***SMS***	***Survey***	***Evaluation***	***Relevant in mobile***
Construction practical consideration	Construction Testing		[2]		X
	Coding	[3], [4], [9]			
	Integration		[2]		
Construction technology	API design and use	[15]		X	X
	Parameterization and Generics	[12]			

Table 4 Testing Activities Refinement

Activity	**Task/Technique**	***SMS***	***Survey***	***Evaluation***	***Relevant in mobile***
Usage based techniques	Thinking aloud	[3] , [12], [16], [9]			X
	Usability test	[3], [4], [9], [11]	[2]	**X**	X
	Field Observation	[3], [13]			X
	Interview	[3], [13],[7],[16], [11]			X
	Satisfaction questionnaire	[3], [16],[14] [9]			X
	Heuristic evaluation	[3], [7], [10],[11]			X
Test Technique	Fault Based technique				

Table 5 Software Management Configuration Activities Refinement

Activity	**Task/Technique**	***SMS***	***Survey***	***Evaluation***	***Relevant in mobile***
Software Release Management and Delivery	Software Release Management		[2]		**X**

Table 6 Maintenance Activities Refinement

Activity	Task/Technique	*SMS*	*Survey*	*Evaluation*	*Relevant in mobile*
Categories of Maintenance	Corrective Maintenance		[2]		
	Perfective Maintenance		[2]		

3.3 Relevance of techniques for the mobile context

For each candidate technique in the first version of the activity catalogue we have considered the relevance for the mobile field. For this purpose, we have analyzed each technique aim, and the way it has been applied by authors of the studies in the SMS, and by developers in our preliminary studies. We have considered the way that the application of every technique contributes to development of a mobile app. With this exercise we have been also able to identify techniques that are difficult to particularize for the mobile context, and therefore we have discarded them in our catalogue.

For every activity type we have analyzed the techniques to be applied, for each one we include a brief description, if it has been included or discarded. We also include how they are particularized for the mobile context for the former, and reason for rejection for the latter.

3.3.1 Requirements: Elicitation

3.3.1.1 User Stories

- **Description**: This technique is commonly used in adaptive methods and refers to short, high level descriptions of required functionality expressed in customer terms.
- **Decision**: Accept.
- **Particularization**: The contexts of use are very important in the use of mobile applications, and the User Stories help to locate the requirements in these contexts of use.

3.3.1.2 Facilitated meeting

- **Description**: The purpose of these meetings is to try to achieve a summative effect, whereby a group of people can bring more insight into their software requirements than by working individually
- **Decision**: Accept.
- **Particularization**: Mobile applications development has certain peculiarities (variety of devices, no continuity of Internet connection, etc.) that the parties involved are not always aware of. The Facilitated Meetings allow to expose these particularities and to be shared by all the stakeholders.

3.3.1.3 Interview

- **Description**: Interviewing stakeholders is a “traditional” means of eliciting requirements. It is important to understand the advantages and limitations of interviews and how they should be conducted.
- **Decision**: Accept.
- **Particularization**: The interviews can be a great help to inquire about what users know about the mobile context and to use it in the development of the new app.

3.3.1.4 Prototype

- **Description**: This technique is a valuable tool for clarifying ambiguous requirements. Low fidelity prototypes are often preferred to avoid stakeholder "anchoring" on minor, incidental characteristics of a higher quality prototype that can limit design flexibility in unintended ways.
- **Decision**: Accept.
- **Particularization**: Low-fidelity prototypes of the apps can be created to elicit the requirements since they present a clear idea of how the interaction between the application's screens and the elements of each screen will be performed. There are applications of mobile prototyping in the market, this demand indicates that it is an interesting technique.

3.3.1.5 Competitor Analysis

- **Description**: Competitor analysis is a technique in which you can take an existing product, and that is, possibly the competence of our application, to perform tests to know the functionalities of the application and interaction techniques.
- **Decision**: Accept.
- **Particularization**: There are many mobile applications available in virtual stores, the same ones that can be downloaded and analyzed. In addition, in the virtual stores, you can obtain additional information about the users' ratings, as well as their comments.

3.3.1.6 Storyboarding

- **Description**: The Storyboards visually present a story scene by scene including the notes of what is happening in each scene.
- **Decision**: Accept.
- **Particularization**: In the mobile field, a storyboard can represent a user's experience in a visual and ordered way when using a mobile application.

3.3.1.7 Survey

- **Description**: Surveys can be a useful tool to obtain specific information when it is not possible to carry out an interview or any other technique that implies a direct relationship with the interested parties.
- **Decision**: Accept.
- **Particularization**: Surveys are very useful in the mobile field, considering that, in general, we do not have a set of defined users. Therefore, it is difficult to make direct contact with a representative group of potential users. Online surveys through the Internet are a good option to obtain information about the features of the new application.

3.3.1.8 Scenarios

- **Description**: Scenarios provide a valuable means for providing context to the elicitation of user requirements. They allow the software engineer to provide a framework for questions about user tasks by permitting "what if" and "how is this done" questions to be asked. The most common type of scenario is the use case description.
- **Decision**: Accept.
- **Particularization**: We can create scenarios that describe the mobile contexts in which users use apps, in order to identify the requirements that are linked to the mobile context.

3.3.1.9 Observation

- **Description**: The importance of software context within the organizational environment has led to the adaptation of observational techniques such as ethnography for requirements elicitation.
- **Decision**: Reject.
- **Justification**: Observation methods are very difficult to carry out, since the usage of mobile apps can be intermittent, maybe with a few minutes dedicated to the app every few hours.

3.3.2 Requirements: Analysis

3.3.2.1 Requirements Classification

- **Description**: Requirements can be classified on a number of dimensions like whether the requirement is functional or nonfunctional, whether the requirement is derived from one or more high-level requirements, according to by requirement priority or according to the scope of the requirement.
- **Decision**: Accept.
- **Particularization**: In the specific case of mobile applications, a dimension can be considered that contemplates the requirements that are directly linked to the mobile domain.

3.3.2.2 Architectural design and requirements allocation

- **Description**: Architectural design is the point at which the requirements process overlaps with software. Despite being a complex task, it is possible to obtain as a result a global structure of the system with its main components, its relationships and how they are distributed.
- **Decision**: Accept.
- **Particularization**: We can include the mechanisms through which mobile elements are included in the architecture of the application, e.g. the way in which interaction with other applications is going to be achieved.

3.3.3 Requirements: Specification

3.3.3.1 System Definition

- **Description**: System definition aims to create a basic document, just showing a high-level description of requirements, since in agile approaches documentation creation is not a priority.
- **Decision**: Reject.
- **Justification**: System definition is a task that cannot be particularized to the mobile field since the task as such is not far from a traditional system definition.

3.3.3.2 Requirements Specification

- **Description**: This document (sometimes known as the user requirements document or concept of operations document) records the system requirements. It defines the high-level system requirements from the domain perspective
- **Decision**: Reject.
- **Justification**: The writing of the requirements is a task that cannot be considered as particular of the mobile field but is a standard task regardless of the scope of the application.

3.3.4 Requirements: Validation

3.3.4.1 Acceptance Test

- **Description**: An essential property of a software requirement is that it should be possible to validate that the finished product satisfies it. Requirements that cannot be validated are really just

"wishes." In most cases, designing acceptance tests does this for how end-users typically conduct business using the system.

- **Decision**: Reject.
- **Justification**: They are not particular to the mobile domain.

3.3.4.2 Prototyping

- **Description**: Prototyping is commonly a means for validating the software engineer's interpretation of the software requirements, as well as for eliciting new requirements. The advantage of prototypes is that they can make it easier to interpret the software engineer's assumptions and, where needed, give useful feedback on why they are wrong.
- **Decision**: Accept.
- **Particularization**: The prototypes can be of great help to validate the requirements elicited, especially those that are directly related to the mobile field, also allow us to assess their understanding by users.

3.3.5 Design: Software structure and architecture

3.3.5.1 Architectural Design Implications

- **Description**: The software architecture provides a high-level abstract description of the structure, behavior and main properties of the system. This abstraction involves the description of the elements with which the system is built and the interaction between them including the mobile ilities.
- **Decision**: Accept.
- **Particularization**: Make explicit in the architecture of the application the mobile ilities that are going to be implemented.

3.3.5.2 Design Patterns

- **Description**: A Design Pattern is a common solution to a common problem in a given context.
- **Decision**: Accept.
- **Particularization**: In the mobile applications field, there are several types of patterns such as those presented by Nilsson [17] or Neil [18].

3.3.5.3 Architectural Styles

- **Description**: An architectural style is a specialization of element and relation types, together with a set of constraints on how they can be used. An architectural style can thus be seen as providing the software's high-level organization. There are various architectural styles of general purpose like client-server, three-tiers, model-view-controller.
- **Decision**: Reject.
- **Justification**: The architectural styles used in mobile applications that have been identified in our studies are the same as those used in desktop or web applications, e.g., the Model-View-Controller.

3.3.6 Design: User interface design

3.3.6.1 User interface design

- **Description**: The interface design process defines how a system interacts with external entities. The design of the user interface defines the way in which users interact with the system and the nature of the inputs and outputs that the system needs. The process to perform the design of the

user interface is iterative, prototypes are often used to represent the characteristics, organization and visual form of the software interface.

- **Decision**: Accept.
- **Particularization**: The process of designing the user interface in the mobile field is one of the important points since it defines the way in which the user interacts with the mobile application. We must pay special attention to the wide variety of devices that exist in the market, each of them has its own characteristics that affect both the performance of the application and how to interact with the user.

3.3.7 Design: Software design notation

3.3.7.1 Structural description

- **Description**: Notations that describe and represent the structural aspects of software design, they are used to describe the major components and how they are interconnected.
- **Decision**: Reject.
- **Justification**: The description of the components through the proposed schemes are at a level of a standard application, making it difficult to make a particularization to the mobile domain.

3.3.7.2 Behavioral description

- **Description**: Notations and languages, some graphical and some textual, that are used to describe the dynamic behavior of software systems and components.
- **Decision**: Reject.
- **Justification**: In the same way to the previous case, the description of the behaviour of the components through the notations and languages are at a level of a standard application, so it is difficult to make a particularization to the mobile domain.

3.3.8 Design: Strategies and Methods

3.3.8.1 Object Oriented Design

- **Description**: Object Oriented Design is a software design paradigm where data and operations are encapsulated in entities called objects.
- **Decision**: Reject.
- **Justification**: It is not a method at the same level as the other methods in this list, it is more a software design approach.

3.3.9 Construction: Construction practical consideration.

3.3.9.1 Coding

- **Description**: Developers write the necessary code to create an app. The considerations to apply in the software construction coding activity are: Techniques for creating understandable source code, use of classes, use of control structures, handling of error conditions, Prevention of code-level security breaches, source code organization, code documentation and code tuning.
- **Decision**: Reject.
- **Justification**: The writing of the code as it is presented in our studies does not differ from the way of writing the code of a web or desktop application.

3.3.9.2 Integration

- **Description**: The purpose of construction testing is to reduce the gap between the time when faults are inserted into the code and the time when those faults are detected, thereby reducing the cost incurred to fix them. Construction involves two forms of testing, which are often performed by the software engineer who wrote the code: Unit testing and Integration testing.
- **Decision**: Reject.
- **Justification**: The integration of the code in a single application is complex to particularize. However, there are authors who consider this activity as part of their proposals, but nothing explicit is identified from the mobile field.

3.3.9.3 Construction Testing

- **Description**: The purpose of construction testing is to reduce the gap between the time when faults are inserted into the code and the time when those faults are detected, thereby reducing the cost incurred to fix them. Construction involves two forms of testing, which are often performed by the software engineer who wrote the code: Unit testing and Integration testing.
- **Decision**: Accept.
- **Particularization**: Muccini et al. [19] consider that for unit tests we can use automatic tools like JUnit in the case of Android, while iOS provides guidelines on how to perform unit tests for mobile applications.

3.3.10 Construction: Technology.

3.3.10.1 API design and use

- **Description**: An application programming interface (API) is the set of signatures that are exported and available to the users of a library or a framework to write their applications.
- **Decision**: Accept.
- **Particularization**: In the mobile field, the APIs are widely used because through them the interaction of the application that is being developed with other applications resident in the same device and with external services is achieved.

3.3.11 Testing: Usage based techniques.

3.3.11.1 Usability Test

- **Description**: The usability test with real users is one of the most common techniques when it comes to testing an application, it provides direct information on how users use the application and what their exact problems are with the specific interface being tested.
- **Decision**: Accept.
- **Particularization**: Although this technique has been widely accepted in traditional development, it should be noted that in the mobile field there must be several aspects such as the place where the test will be carried out, the devices and the platforms to be used. In addition, we must consider the parameters to measure the relationships with mobile ilities. The data of tasks performed by participants in a mobile device is recorded to be analyzed, a diagnosis of real problems and recommendations to solve such problems is made.

3.3.11.2 Thinking aloud

- **Description**: In thinking aloud protocol, participants are asked to talking while they are doing their task. By verbalizing their thoughts, users allow us to understand how they see the application, and this facilitates the identification of the main problems of users.
- **Decision**: Accept.
- **Particularization**: In the mobile field, this technique can be applied to know the user's impression when performing a task and allows us to know the perception that the user has of the mobile application and what are the difficulties that are linked to the mobile domain.

3.3.11.3 Field Observation

- **Description**: During the evaluation, the observation is performed with the user doing a task, but with the prototype or the application to identify possible usability problems.
- **Decision**: Reject.
- **Justification**: It is very complicated and expensive to do if the mobile application is used internally.

3.3.11.4 Survey

- **Description**: Surveys like evaluation tools are used to obtain specific information about the usage of systems.
- **Decision**: Accept.
- **Particularization**: Surveys are can be useful in the mobile field, to know what the perception of the performance and usage of our app in a group of users is. Online surveys through the Internet are a good option to obtain this information

3.3.11.5 Satisfaction Questionnaire

- **Description**: The questionnaires allow knowing a subjective impression of the participants in a usability test.
- **Decision**: Accept.
- **Particularization**: This technique allows knowing the impression of users when using the developed mobile application.

3.3.11.6 Heuristic Evaluation

- **Description**: Heuristic evaluation is a method for structuring the critique of a system using a set of relatively simple and general heuristics. Heuristic evaluation involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles.
- **Decision**: Accept.
- **Particularization**: The set of heuristics that are often used are those proposed by Nielsen et al. [20]. However, in the mobile context, not all heuristics are applicable, this is how Bertini et al. [21] propose a list of heuristics that can be applied to mobile devices. Nor does it rule out the existence of new heuristics that can be applied to the mobile context.

3.3.12 Software Configuration Management: Software Release Management and Delivery.

3.3.12.1 Software Release Management

- **Description**: Software release management encompasses the identification, packaging, and delivery of the elements of a product. For example, an executable program, documentation, release notes, and configuration data.
- **Decision**: Accept.
- **Particularization**: All the required elements must be prepared to launch the application to the virtual store of each platform. Each platform has its own particular requirements to accept an application or an update. This task is alienated to the continuous delivery approach (Continuous Delivery) that is applied by the development teams to produce software in short periods of time ensuring that the software update is available at any time.

3.3.13 Maintenance

3.3.13.1 Corrective Maintenance

- **Description**: Corrective maintenance is a reactive modification (or repairs) of a software product performed after delivery to correct discovered problems. Included in this category is emergency maintenance, which is an unscheduled modification performed to temporarily keep a software product operational pending corrective maintenance.
- **Decision**: Reject.
- **Justification**: The repair of the application is not a process that presents any particular task that is different from those performed in software engineering.

3.3.13.2 Perfective Maintenance

- **Description**: Perfective maintenance is a modification of a software product after delivery to provide enhancements for users, improvement of program documentation, and recoding to improve software performance, maintainability, or other software attributes.
- **Decision**: Reject.
- **Justification**: Modifying the application like the previous point is not a process that presents any particular task that is different from the one performed in software engineering.

4 Conclusions

A proposal for a catalogue of development activities for the app has been prepared, taking as a starting point the information obtained from empirical studies on the development of apps. The resulting catalogue contains both development activities and a set of tasks/techniques that can be applied in the development of apps, without being prescriptive, we consider it a contribution for developers in an area that is constantly growing and where there is a lack of information about process issues.

5 References

[1] IEEE, 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology., IEEE, 1990. doi:10.1109/IEEESTD.1990.101064.

[2] L. Chandi, C. Silva, D. Martínez, T. Gualotuña, Mobile application development process: A practical experience, in: Á. Rocha, B. Alturas, C. Costa, L.P. Reis, Manuel Pérez Cota (Eds.), 2017 12th Iber. Conf. Inf. Syst. Technol., Information Systems and Technologies (CISTI), 2017 12th

Iberian Conference on, Lisboa, 2017: pp. 2113–2118. doi:10.23919/CISTI.2017.7975825.

[3] B. Losada, M. Urretavizcaya, J.-M. López, I. Fernández-Castro, Applying usability engineering in InterMod agile development methodology. A case study in a mobile application, J. Univers. Comput. Sci. 19 (2013) 1046–1065. http://www.scopus.com/inward/record.url?eid=2-s2.0-84882957947&partnerID=40&md5=a3c7ac82f5c94448a986b0993dcff202.

[4] A. Hameed, A. Oudah, Improved methodology for mobile commerce applications, Int. J. Softw. Eng. Its Appl. 8 (2014) 29–42. doi:10.14257/ijseia.2014.8.8,04.

[5] C. Scharff, R. Verma, Scrum to support mobile application development projects in a just-in-time learning context, in: Proc. 2010 ICSE Work. Coop. Hum. Asp. Softw. Eng. - CHASE '10, ACM Press, New York, New York, USA, 2010: pp. 25–31. doi:10.1145/1833310.1833315.

[6] C.E. de A. Freire, M. Painho, Development of a Mobile Mapping Solution for Spatial Data Collection Using Open-Source Technologies, Procedia Technol. 16 (2014) 481–490. doi:10.1016/j.protcy.2014.10.115.

[7] T. Siu, V. Herskovic, Mobile augmented reality and context-awareness for firefighters, IEEE Lat. Am. Trans. 12 (2014) 42–47. doi:10.1109/TLA.2014.6716491.

[8] V. Vylegzhanina, D.C. Schmidt, P. Hull, J.S. Emerson, Q. M.E., S. Mulvaney, Helping children eat well via mobile software technologies, 2nd Int. Work. Mob. Dev. Lifecycle, MobileDeLi 2014. (2014) 9–16. doi:10.1145/2688412.2688413.

[9] F. Ferreira, N. Almeida, A.F. Rosa, A. Oliveira, J. Casimiro, S. Silva, A. Teixeira, Elderly Centered Design for Interaction – The Case of the S4S Medication Assistant, Procedia Comput. Sci. 27 (2014) 398–408. doi:10.1016/j.procs.2014.02.044.

[10] W. Woods, E. Scanlon, iSpot Mobile - A Natural History Participatory Science Application Conference Item Application, (2012).

[11] E. Villalba, D. Salvi, M. Ottaviano, I. Peinado, M.T. Arredondo, A. Akay, Wearable and Mobile System to Manage Remotely Heart Failure, Circulation. 13 (2009) 1386–1397. doi:10.1161/CIRCULATIONAHA.108.802918.

[12] P. Mata, A. Chamney, G. Viner, D. Archibald, L. Peyton, A development framework for mobile healthcare monitoring apps, Pers. Ubiquitous Comput. 19 (2015) 623–633. doi:10.1007/s00779-015-0849-9.

[13] R. Skiada, E. Soroniati, A. Gardeli, D. Zissis, EasyLexia: A Mobile Application for Children with Learning Difficulties, Procedia Comput. Sci. 27 (2014) 218–228. doi:10.1016/j.procs.2014.02.025.

[14] A. Kinai, R.E. Bryant, A. Walcott-Bryant, E. Mibuari, K. Weldemariam, O. Stewart, Twende-twende: a mobile application for traffic congestion awareness and routing, in: Proc. 1st Int. Conf. Mob. Softw. Eng. Syst. - MOBILESoft 2014, ACM Press, New York, New York, USA, 2014: pp. 93–98. doi:10.1145/2593902.2593926.

[15] M. Antic, Development of eStudent iOS mobile application, IJIM. 7 (2013) 35–40.

[16] B. Peischl, M. Ferk, A. Holzinger, The fine art of user-centered software development, Softw. Qual. J. 23 (2015) 509–536. doi:10.1007/s11219-014-9239-1.

[17] E.G. Nilsson, Design patterns for user interface for mobile applications, Adv. Eng. Softw. 40 (2009) 1318–1328. doi:10.1016/j.advengsoft.2009.01.017.

[18] T. Neil, J. Tidwell, Mobile design pattern gallery : UI patterns for smartphone apps, n.d.

[19] H. Muccini, A. Di Francesco, P. Esposito, Software Testing of Mobile Applications: Challenges and Future Research Directions, in: AST '12 Proc. 7th Int. Work. Autom. Softw. Test, Zurich, 2012: pp. 29–35. https://s3.amazonaws.com/academia.edu.documents/43597250/AST2012_TestingMobileApps_2.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1529486284&Signature=UzJ4ol0if1aWTEIuQrUYs7Fx2TA%3D&response-content-disposition=inline%3B filename%3DSoftware_Testing_of_ (accessed June 20, 2018).

[20] J. Nielsen, R. Molich, Heuristic Evaluation of user interfaces, CHI '90 Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (1990) 249–256. doi:10.1145/97243.97281.

[21] E. Bertini, S. Gabrielli, S. Kimani, Appropriating and assessing heuristics for mobile computing, in: Proc. Work. Conf. Adv. Vis. Interfaces - AVI '06, 2006: p. 119. doi:10.1145/1133265.1133291.