A Testability and Observability Framework to Assure Traceability Requirements on System of Systems

Leticia Morales Trujillo[1,*], Miguel Ángel Olivero González[1,2], Francisco José Domínguez Mayo[1], Julián Alberto García García[1] and Manuel Mejías Risoto[1]

[1]*Department of Languages and Computer Systems, Web Engineering and Early Testing (IWT2) group, University of Seville, Spain*
[2]*Istituto di Scienza e Tecnologiedell'Informazione, ConsiglioNazionaledellaRicerche, Italy*
E-mail: leticia.morales@iwt2.org; miguel.olivero@iwt2.org; fjdominguez@us.es; juliangg@us.es; risoto@us.es
[*]*Corresponding Author*

Received 20 December 2019; Accepted 14 April 2020;
Publication 03 June 2020

Abstract

The advance in the digital world has caused a growth of complexity in innovation. Traditional approaches to innovation, based on reductionism, face greater difficulties. That is why we have witnessed the growth of those known as System of Systems (SoS). There is a wide variety of methodologies and domains of application in the literature to form framed solutions in the context of SoS, but there is no unified consensus for its use and even less when it comes to agile environments of continuous integration and deployment in which traceability requirements are critical. In recent years, the need to have traceability software that continuously records and monitors the trace of the entities that interact with it has become an essential feature. In addition, over the years there has been evidence of errors caused by poor traceability

Journal of Web Engineering, Vol. 19_2, 297–318.
doi: 10.13052/jwe1540-9589.1928

control. Therefore, this document presents an agile framework that aims to guarantee the traceability of a SoS from the early stages. This framework unifies the discovery, development and operations, providing full coverage in the conformation of the solution. Finally, we present a case study as future work, which is based on the application of our framework on smart laboratories for assisted reproduction.

Keywords: System of Systems (SoS), traceability, framework.

1 Introduction

Today, progress in the digital world implies that technological products and systems are increasingly interdependent, leading to increasing complexity in design and challenges for innovation. Due to the increasing complexity, traditional innovation approaches based on reductionism, which focuses on the decomposition of the system and the optimization of subsystems, components and their interrelationships within existing products or systems, face greater difficulties in the search of future innovation opportunities That is why, we have witnessed the growth of those known as System of Systems (SoS) [1–4].

SoS are a collection of dedicated or task-oriented systems that combine their resources and capabilities to create a new and more complex system that offers more functionality and performance than simply the sum of the constituent systems. They are created through a process of synthesis of previously existing, but not related, independent systems [5–9]. This synthesis creates value resulting in the new system improving the utilities or functionalities of each of the previous systems or enabling novel functions of the new holistic system. Some examples of Systems of Systems are the management of air traffic, the European railway network, integrated land transport, emergency service and personal health management, the means of communication, among many others [10, 11].

System of systems engineering (SoSE) is the set of processes, tools and development methods to design, redesign and implement solutions to system challenges [12, 13]. It addresses the design, development and operations of evolving programs. That is, it seeks to optimize the network of several legacy interactive systems and new systems together to meet multiple objectives of a program.

An effective software development methodology in SoSE should prepare decision makers to design solutions for SoS problems. Due to the wide variety

of methodologies and application domains present in the literature, there is no single unified consensus for the processes involved in Systems and Systems Engineering [14].

In many sectors, it is essential to have traceability software that continuously records and monitors the trace of the entities that interact with it. These entities can be objects, actors or activities. If we also want to design a software solution that guarantees compliance with traceability in an SoS environment, the complexity increases even more. We understand traceability as a set of measures, actions and procedures that allow registering and identifying an entity from its origin to its final destination [15, 16].

Over the years there have been several errors regarding poor traceability control; in the field of medicine in general [17], in the field of food [18, 19], in the field of assisted reproduction [20–23], among many others.

Therefore, it is intended to develop a solution that integrates a smart contract (contract between two or more parties, which is capable of executing and enforcing itself, autonomously and automatically) where each and every traceability requirement identified in an SoS environment is collected, so that entities can register and control themselves continuously in the value chain. All this to guarantee in the future the control and monitoring of the entities that interact with the systems, so that the risk of error is reduced to minimum levels, making use of information technologies as the central axis of the solution and increasing the trust of those involved.

To achieve this objective, this document presents an agile methodology that aims to ensure the traceability of an SoS from the early stages, although it may be applicable to simple systems. This framework unifies the discovery, development and operations, which implies full coverage in the conformation of the solution. This framework will, in the future, be applied in a smart laboratorio for assisted reproduction. This case study is presented in this article as future work.

The present work is structured as follows; In section 2, the background of the work and the main objectives that are to be achieved both with this work and in the future are presented. In section 3, the proposed framework for the generation of software solutions that guarantee traceability is presented. In section 4, related works are exposed, that is, publications that consist of methodologies of discovery, development and operations, already existing in the literature. Section 5 shows the general conclusions of the work. Finally, in section 6, open lines of work are broadly discussed.

2 Background and Objectives

Over the years there have been several errors regarding poor traceability control; in the field of medicine in general, as the case revealed in a study conducted by the College of American Pathologists in which the tissues received from a patient in one or more containers, corresponding to the same procedure, were assigned the same number of identification of access to admission in the pathology department [17]. In the food sector, as in 1999 when Coca-Cola recognized that its soft drinks in Belgium were contaminated due to a fungicidal treatment, but cases of poisoning were detected in Belgium, Holland and Luxembourg [18] or as in the case of contaminated meat from listeriosis in Spain in 2019 [19].

In the field of assisted reproduction, human reproduction laboratories are a clear example in which traceability should be known. It is necessary to know at all times the trace followed by biological samples and test. In addiction, human reproduction laboratories are within the context of SoS since there are a large number of interconnected systems. As mentioned earlier, SoS are a collection of dedicated or task-oriented systems that combine their resources and capabilities to create a new and more complex system that offers more functionality and performance than simply the sum of the constituent systems. Sample management is a critical aspect that requires all appropriate mechanisms to ensure traceability continuously, avoiding fatal errors.

In the Netherlands in 1993, a white woman gave birth to a black child and a white child after receiving "mixed sperm from a poorly sterilized pipette" [20]; in the United States in 1998, a white woman white gave birth to a black baby in what became known as the case of "scrambled eggs". After a "bitter battle for custody", the black couple whose embryo was implanted by mistake in the white woman won custody of the black baby [21]; in October 2002 there was another confusion of IVF (in vitro fertilization) in Britain. In April, two women received the "wrong embryos" in a confusion that involved three women. One woman had her own "worst quality embryos" implanted, while her "best quality partner" was assigned to another woman whose embryos were mistakenly sent to a third woman. The women who received the "wrong embryos" were "devastated" and "traumatized" after undergoing an "emergency procedure to remove the embryos" [22]; in 2016, the University Hospital of Utrecht made public the possible fertilization of oocytes of 26 women with sperm outside their partner, that is, about twenty women or couples have fathered children with the sperm of the man who was not indicated. Finally, the center detected that the pipette that had been used

in some oocyte fertilization procedures was contaminated with the sperm of another patient [23].

The traceability of a system translates as a series of requirements that must be met. The requirements of a system arise from the needs of the client, from the limitations of the environment where it is going to be implemented or from the management of the information that the system must perform. Normally they will represent values that must meet at least or at most each one of the developed aspects. They serve to limit the functionality or construction of the system, assuming limits for the design and listing all the functionalities it should cover.

There are several types of requirements encompassed in two large groups: (i) functional requirements that directly affect the main functionality of the system and (ii) non-functional requirements, which do not represent the main functionality of the system, but impose design or implementation restrictions. They are properties or qualities that the system must have.

Traceability requirements are positioned within the group of non-functional requirements and include all system actions, such as: actions of which the traces (track), the format and the storage medium of each type of trace should be stored, the management of the expiration of the traces and their history, control and means of access to the traces of the system, etc. Traceability could be defined as the actions that allow the monitoring of different types of elements, through continuous monitoring of physical parameters and thanks to the use of new technologies [24]. Traceability not only covers the basic requirements that products can be traced along the value chain, but also the possibility of specifying what they are made of and how they have been processed.

To register the agreements made with a client during the development of the system, the contracts are used. A contract is nothing more than an agreement between two or more parties, an environment that defines what can be done, how it can be done, what happens if something is not done, etc. That is, some rules that allow parties that accept it to understand what the interaction they will carry out will consist of [25–27].

So far, the contracts have been verbal documents or expensive documents, subject to laws and jurisdictions that sometimes require notaries. That is, more costs, time and third parties involved in the process. Because of this, they are not accessible to anyone. Instead, a smart contract is a computer program that facilitates, ensures, enforces and executes registered agreements between two or more parties.

Smart contracts aim to provide superior security to traditional contract law and reduce transaction costs associated with contracting, since they are able to be maintained and enforced, autonomously and automatically, without intermediaries or mediators.

For all the above, it is intended to develop a solution that integrates a smart contract where each and every traceability requirement identified in an SoS environment is collected, so that entities can register and continuously monitor themselves in the value chain

In this context, the objective of this work is to propose an agile framework, that is, to promote the continuous iteration of the software development life cycle, framed within the context of System of Systems engineering, to achieve the solution.

3 Proposed Framework

As mentioned above, our proposed framework is focused to guarantee the traceability of a software solution framed in the context of SoS from the early stages. That is, it ensures the continuous recording and control of the interaction of different types of entities with a system, which allows monitoring. To ensure traceability from the early stages it is necessary: to validate, to verify and to monitor the registration and control of the solution that will be created (Figure 1).

- Validate. Confirm that what is being done is being done well and the intended objective is achieved. It helps us to know, a future, the registration and control of the entities is done well and the necessary is recorded and monitored.
- Verify. Check that what is said is really done. It helps us to verify, a past, what is supposed to be done, it is and works as expected.
- Monitor. Control the development of an action or an event through one or more monitors. It helps us to know, in the present, the registration and control of all entities.

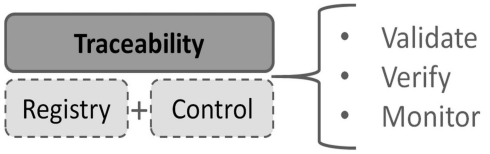

Figure 1 Traceability concept and how to guarantee it.

These actions (validate, verify and monitor) are carried out throughout the proposed framework in a preventive manner, that is, taking precautions or measures in advance to avoid risks, and in a corrective manner, making the necessary modifications to eliminate errors.

This methodological proposal is based on empirical methodologies and it consists of three main stages: discovery (DIS), development (DEV) and operations (OPS). On the one hand, in the discovery stage (Section 3.1) a series of sub-stages proposed by the Design Thinking [28] process is carried out and it is where the validation of the registration and the control of the solution to be carried out is carried out. On the other hand, both the development stage (Section 3.2) and the operations stage (Section 3.3) proposed here follow the practices established in DevOps [29–31] and the workflow of GitFlow [32–34], and it is where the verification and monitoring of the registry and control of the solution to be created is carried out.

As mentioned earlier, the proposed framework is within the context of the agile development methodology. Agile development methodologies are the methodologies that promote continuous iteration of the development life cycle and that follow a series of agile principles and values.

Figure 2 shows the continuous iteration that takes place between the general process and between the different stages. New approaches are continually being proposed in the discovery stage, in turn other approaches are being developed in the development stage and the offering a service at the stage of operations.

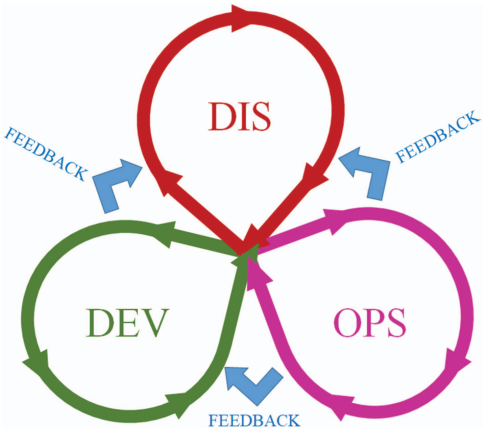

Figure 2 Iteration in the methodological proposal.

In addition, feedback arrows can be visualized, which help to confirm that everything is done as planned and that the objectives set (DEV → DIS) are achieved, to verify that the planned one is actually done (OPS → DEV) and to control what It is provided as a service is necessary (OPS → DIS).

In this case, to ensure traceability, it is very important to validate, verify and continuously monitor the registration and control of the entities that interact with the system. These actions are typical of the figure of the tester, so the principles and values that are followed in this framework are those proposed in Agile Testing [35]:

- Building quality in: teams focus on preventing misunderstandings about features behavior as well as preventing defects in the code.
- Guiding development with concrete examples: using practices likes acceptance test-driven development (ATDD), behavior-driven development (BDD), or specification by example (SBE).
- Including testing activities such as having conversations to build shared understanding; asking questions to test ideas and assumptions; automating tests; performing exploratory testing; testing for quality attributes like performance, reliability, and security; and learning from production usage.
- Using whole-team retrospectives and small experiment to continually improve testing and quality and find what works in their context.

As indicated throughout the work, the proposed framework consists of three main stages (discovery, development and operation), which in turn are formed by a series of sub-stages. Figure 3 shows the framework structure in general, that is, the stages and substages of which they are composed are shown. Regardless of the iterations between the different stages and substages (Figure 2). It is also indicated by points, the sub-stages where the validation, verification and monitoring actions are carried out, both preventive and corrective.

3.1 Discovery

Its objective is to achieve knowledge and understanding of a problem in a given domain, propose a solution and obtain as a result a solution proposal validated by the end user. To achieve this goal, a series of stages proposed by the Design Thinking process are carried out [28]:

- Empathize. This stage aims to meet the public to whom the efforts will be directed.

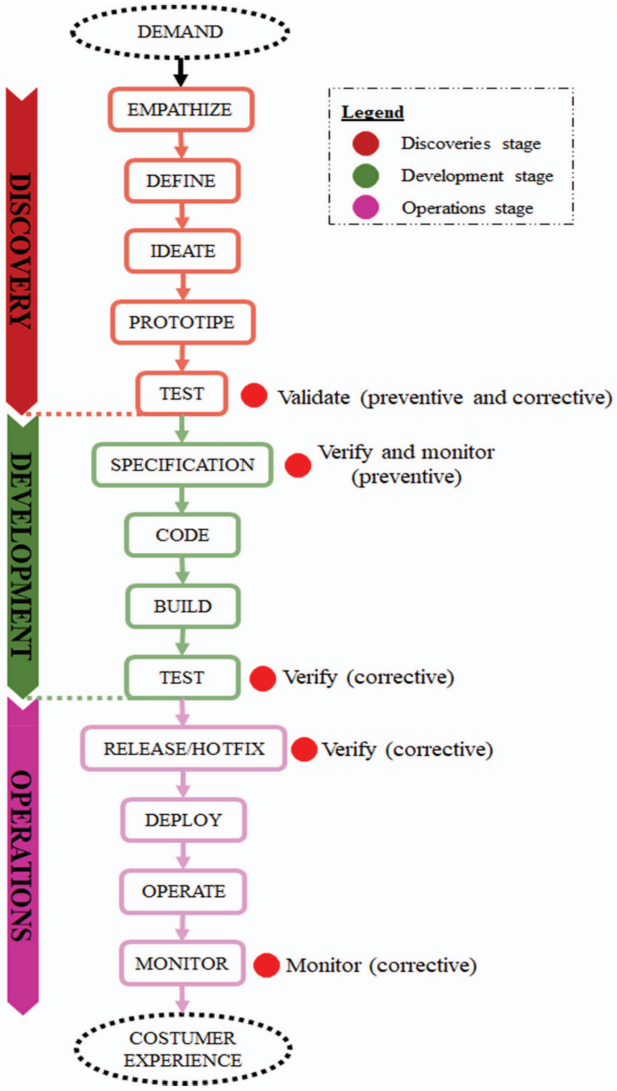

Figure 3 Proposed framework.

- Define. All the information collected in the previous stage allows you to specify one or several improvement opportunities. It will be the team's decision to prioritize which one will be attacked first and which in subsequent actions.

- Ideate. Once the information has been analyzed and the problems are defined according to the users, the ideas that will be filtered later will be generated to prioritize the most plausible ones.
- Prototype. The purpose of this stage is to create realistic and economical versions of the product or service, where the ideas of the previous stage are applied.
- Test. Prototypes are used to test with users. The conclusions obtained from them allow to iterate, that is to say: empathize even more, perfect ideas, create prototypes again and try again to obtain solutions that really respond correctly to the problems of the users. The validation of said prototype is carried out with simulations [36], so that the end user can validate that all the necessary traceability requirements are recorded through continuous monitoring (preventive and corrective validation).

At this stage the validated solution prototype would be obtained, with all the traceability requirements that must be met in the SoS environment.

3.2 Development

The objective of this stage is to obtain tangible and validated results of the proposed solution. To achieve this goal, a series of stages are carried out:

- Specification. It is the task of describing the software in detail. The specification is made in sufficient detail so that qualified developers can develop the solution with minimal additional effort. In this task, all the requirements identified in the previous stage are defined in detail using class diagrams, business diagrams, pseudocode, etc. defined by widely used standards, such as those proposed by the OMG (Object Management Group). Among other functionalities, it is specified how the monitoring will be carried out, that is, the visualization and control of the registered entities (preventive monitoring). In addition, the tests are defined following good practices defined by agile tests (Technology-facing tests that guide development, business-facing test that guide development, business-facing test that critique the product and technology-facing test that critique the product) [35] that will be carried out after the solution is developed to verify that everything is registered and controlled as planned (preventive verification).
- Code. It is the task of translating the specification into code and performing relevant tests in the developed module.
- Build. Action to integrate the implementation of the code generate in the development environment.

- Test. Action to verify that the software responds/performs correctly the tasks indicated in the specification, that is, to verify that what has been specified is recorded and to control compliance with the provisions (corrective verification). Based on the conclusions obtained from the defined tests, iterations are performed to obtain solutions that really respond to what is specified.

This stage is where the tangible software solution is obtained, validated and verified based on what was agreed with the end user and is integrated into the SoS context, but is not yet available for end user use because it is still in the development environment.

3.3 Operation

This stage aims to ensure the correct functioning of the solution in production environments once implemented. This stage covers everything related to the functions of IT (Information Technology) that are not related to the development and management of the application. To achieve this goal, a series of task are carried out:

- Release/Hotfix. Deployment in the pre-production environment and conducting the relevant tests to verify that what has been specified is recorded and monitor compliance with the provisions (corrective verification). According to the results obtained in the tests, there are two options: successful result, implementation in the production environment, failed result, return to the development environment and error correction, as proposed in the Gitflow workflow [32].
- Deploy. Task deployment in the production environment.
- Operate. The product is operational and the end user is already using it.
- Monitor. Continuous monitoring task of aspects of the software solution (corrective monitoring). The information that should be monitored and how it should behave is previously defined in the stages of discovery and development.

This stage is where the tangible software solution is obtained, validated, verified and monitored based on what was agreed with the end user, integrated in the SoS context, available and used by the end user.

4 Related Work

Along this work it has been described a methodology that guarantee data traceability in a SoS context from early stages as a combination of other

methodologies. There are different methodologies and good practices to aid in the process of capturing the software requirements, its development, its testing, and its monitoring. In particular, this approach is inspired in (1) the User-centered Design methodologies to capture the requirements and (2) Agile development methodologies.

User-centered design methodologies are a set of methods and techniques with the purpose of know and understand the users' needs, limitations, behavior and nature [42]. When applying such methodologies, potential or even real final users are involved. The interaction with such potential final users are carried out in a iterative-incremental process. On each incremental round the software specifications are increasingly growing and fitting to final user expectations. Design Thinking [28] or Design Sprint [43, 44] are two well-known approaches among user-centered design methodologies. In particular, the Design Sprint techniques has been successfully applied in previous work in real industrial context in the healthcare context to define the software specifications [45]. According to those previous experiences the use of these methodologies, that consider involving final users in early definition stages, produce a software specification that are less susceptible to modifications in late stages in the software lifecycle.

The agile development methodologies are a set of operating guidelines under the agile manifesto recommendations [46]. Agile methodologies are those methodologies that meet certain requirements as described in the manifesto. Such methodologies are Software Engineering methods based in a incremental and iterative development. The software is developed in self-organized and multidisciplinary teams [47]. DevOps is one of the most common sets of good agile development practices that includes continuous implementation and integration [48]. Different studies have been conducted regarding the DevOps practice applied in the industry and in the academia [49, 50]. Other researches, as the one conducted by Virmani, brought closer continuous integration and continuous delivery [51].

To the best of our knowledge, there are not approaches considering an user-centered design strategy explicitly involving the the agile manifesto.

The traceability requirements are non-functional requirements that have been studied since more than two decades ago. Studies dealing with traceability aim to describe the behavior of a system at the time of attending the changes on the data and functionalities [39]. Different approaches have been conducted to enhance traceability requirements in ordinary systems. Such requirements usually consider, systems modelling, and techniques for better

applying [40, 41]. Notwithstanding it is still a work in progress in the SoS context.

The SoS is an emerging field and not many methodologies may be found in the literature for these systems constructions. Authors have been focusing the most in development methodologies among the already existing ones applicables in SoS. DeLaurentis, for instance proposed a method on which a SoS is abstracted, modelled, and analyzed in a method to detect behavior patterns [14]. Mittal and Risco-Martin have proposed an unified development process across different constituent systems [37]. Regarding non-functional requirements in the SoS, the interest for security is growing. In recent years it is being considered as a first-class feature, which promote new emerging researches as the TeSSoS approach [38].

This work combine user-centered design, and agile manifesto perspectives and tackles the traceability requirements in a user-centered agile methodology. To characterize the relevance of traceability when sharing data, the proposed methodology is designed by focusing in the SoS context.

5 Conclusions

The development of this work has involved the immersion in the depth of the processes carried out in the methodologies of discovery, development and agile operations in System of Systems contexts. Through the search for a solution to solve the existing problem, referring to poor traceability control, knowledge about the technologies being investigated and those that currently exist is expanded. This work arises from this fundamental idea: *how can control of the traceability of the entities be increased within a System of Systems context?*

More and more sectors in which it is essential to have traceability software that continuously records and monitors the trace of the entities that interact with it and due to the progress in the digital world, products and systems are increasingly interoperable and this makes Make your design more complex. This interoperability between products and systems is what is known as a System of Systems context.

To all this, the errors that refer to the poor control of traceability that has been recorded over the years are added.

That is why in the future it is intended to develop a solution that integrates a smart contract where each and every traceability requirement identified in an SoS environment is compiled. In this way, entities can register and control along the value line.

To achieve this objective, this document presents an agile framework framed in the context of Systems of Systems Engineering.

Within this methodological framework, the discovery stage, which aims to achieve knowledge and understanding of the problem in a given domain, propose a solution and obtain a solution proposal validated by the end user; the development stage, whose objective is to obtain tangible and validated results of the proposed solution; and the operation stage, whose objective is to guarantee the correct functioning of the solution in production environments once implemented, continuously monitoring the interaction of the entities with the systems, are unified.

The unification of the stages of discovery, development and operations implies a total coverage of the conformation of the solution.

Each of the stages of the proposed framework, in turn, is composed of a series of sub-stages.

As it is an agile framework, continuous iteration between its stages is promoted and within each of the stages, continuous iteration between the sub-stages is promoted and a series of agile principles and values are followed.

6 Future Works

This work is about the beginning of a doctoral thesis work. The final objective of the doctoral thesis is to develop a solution that integrates a smart contract where each and every traceability requirement identified in an SoS environment is collected, so that entities can register and continuously monitor in the value chain.

The following lines of work proposed to expand this research begin in greater detail in each of the stages and substages of which the framework are composed, validating the proposed framework in a real context. Which implies, the performance of all the tasks related to the discovery, all the tasks related to the development and all the tasks related to the operation. Never forgetting that we are in a context of System of Systems where the traceability of the entities that interact with this solution must be recorded and monitored continuously.

The context of the application will be framed in the commercial domain of an assisted reproduction clinic, more specifically, in an intelligent reproduction laboratory.

Assisted reproduction has become a clinical service that more and more people access. Current problems such as the delay in paternity age, single

parents, etc. options and the different treatments that are put at the service of society have proliferated. A fundamental part of these processes lies in laboratory work.

Human reproduction laboratories are exposed to multiple incidents, but one of the most serious is the one that causes the incorrect identification of biological samples (eggs, sperm and embryos), being incorrect identification a common problem in all areas of health.

An identification error occurs when a patient is incorrectly combined with a test, sample, treatment or procedure and is usually caused by stress, work overload, multiple interruptions, material errors, among many factors.

Certain processes, such as the mixing of ovules and sperm, and the transfer of embryos to the uterus, are considered critical, as they represent "the point of no return." If in the laboratories of assisted reproduction an erroneous identification occurs, it can go unnoticed practically in each of the steps of the process that involves gametes and embryos. The final result will be catastrophic for the patient, the professional and the clinic, with legal implications that can lead to sanctions and even, in extreme cases, to the closure of the clinic.

For all this, it is necessary to know the traceability that follows the biological samples and patient samples at all times within the laboratory. In an intelligent laboratory, the interrelation between different systems is essential, so we will find ourselves in an environment in the context of SoS.

When validating a proposal, we must always keep in mind that we can find some threats. In this case, the greatest dawn we can face when validating the proposed framework is when collecting information to define traceability requirements.

As discussed throughout the document, the traceability requirements are what will allow me to register and control at all times the entities that interact with the different systems. Currently, it is not an extended practice among software engineers to collect specific information of this type and, therefore, customers are not accustomed to providing such information. It is not known what is the best and most productive way to collect information of this type.

On the other hand, an exhaustive study will be carried out to continue this work and provide a clear explanation of the current state of traceability in Systems of Systems environments.

Finally, once we ensure that the framework is applicable, it will be extended to other application contexts.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (POLOLAS, TIN 2016-76956-C3-2-R) and by the GAUSS national research project (MIUR, PRIN 2015, Contract 2015KWREMX).

References

[1] Popper, S. W., Bankes, S. C., Callaway, R., &DeLaurentis, D. (2004). System of systems symposium: Report on a summer conversation. Potomac Institute for Policy Studies, Arlington, VA, 320.

[2] Ameri, F., Summers, J. D., Mocko, G. M., & Porter, M. (2008). Engineering design complexity: An experimental study of methods and measures. Res. Eng. Des, 19(2–3), 161–179.

[3] De Weck, O. L., Roos, D., & Magee, C. L. (2011). Engineering systems: Meeting human needs in a complex technological world. Mit Press.

[4] Luo, J., & Wood, K. L. (2017). The growing complexity in invention process. Research in Engineering Design, 28(4), 421–435.

[5] Boardman, J., & Sauser, B. (2006, April). System of Systems-the meaning of of. In 2006 IEEE/SMC International Conference on System of Systems Engineering (pp. 6-pp). IEEE.

[6] DeLaurentis, D. (2007, July). Role of humans in complexity of a system-of-systems. In International Conference on Digital Human Modeling (pp. 363–371). Springer, Berlin, Heidelberg.

[7] Eisner, H., Marciniak, J., & McMillan, R. (1991, October). Computer-aided system of systems (S2) engineering. In Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics (pp. 531–537). IEEE.

[8] Jamshidi, M. O. (2008). System of systems engineering-New challenges for the 21st century. IEEE Aerospace and Electronic Systems Magazine, 23(5), 4–19.

[9] Keating, C., Rogers, R., Unal, R., Dryer, D., Sousa-Poza, A., Safford, R., ... & Rabadi, G. (2003). System of systems engineering. Engineering Management Journal, 15(3), 36–45.

[10] Feng Tian, "An agri-food supply chain traceability system for China based on RFID &blockchain technology," 2016 13th International Conference on Service Systems and Service Management (ICSSSM), Kunming, 2016, pp. 1–6.

[11] Systems of Systems (SoS). (2019). https://www.sebokwiki.org/wiki/Systems_of_Systems_(SoS) (accessed February 21, 2020)

[12] Cureton, K. L., & Settles, F. S. (2005, October). Systems-of-systems architecting: educational findings and implications. In 2005 IEEE International Conference on Systems, Man and Cybernetics (Vol. 3, pp. 2726–2731). IEEE.

[13] Keating, C., Rogers, R., Unal, R., Dryer, D., Sousa-Poza, A., Safford, R., ... &Rabadi, G. (2003). System of systems engineering. Engineering Management Journal, 15(3), 36–45.

[14] DeLaurentis, D., Sindiy, O., & Stein, W. (2006). Developing Sustainable Space Exploration via System-of-Systems Approach. In Space 2006 (p. 7248). (accessed December 19, 2019)

[15] TRAZABILIDAD. Según el Comité de SeguridadAlimentaria de AECOC. (2016) https://docplayer.es/9988201-Trazabilidad-segun-el-comite-de-seguridad-alimentaria-de-aecoc.html (accessed February 21, 2020)

[16] Trazabilidad. (2017) https://trazabilidadalimentaria.blogspot.com/ (accessed February 21, 2020)

[17] Nakhleh, R. E., Idowu, M. O., Souers, R. J., Meier, F. A., &Bekeris, L. G. (2011). Mislabeling of cases, specimens, blocks, and slides: a College of American Pathologists study of 136 institutions. Archives of pathology & laboratory medicine, 135(8), 969–974.

[18] MECALUX ESMENA, El rastroconfuso de la trazabilidad. (2005) . https://www.mecalux.es/articulos-de-logistica/rastro-confuso-trazabilidad

[19] VITÓNICA, Trazabilidad y alertasalimentarias: qué ha podidosalir mal en el recientecaso de listeriosis. (2019). https://www.vitonica.com/prevencion/trazabilidad-alertas-alimentarias-que-ha-podido-salir-mal-reciente-caso-listeriosis (accessed December 19, 2019)

[20] Spriggs, M. (2003). IVF mixup: white couple have black babies. Journal of medical ethics, 29(2), 65–65.

[21] Dyer, O. (2002). Black twins are born to white parents after infertility treatment. BMJ, 325(7355), 64.

[22] BBC NEWS, Embryo mix-up at IVF hospital, (n.d.). http://news.bbc.co.uk/2/hi/health/2367705.stm (accessed December 19, 2019).

[23] EL PAÍS, Holandainvestiga la posiblefecundación de 26 mujeres con esperma equivocado |Internacional, (n.d.). https://elpais.com/internacional/2016/12/29/actualidad/1483021366_741815.html (accessed December 19, 2019).

[24] INDRA, Trazabilidaden la cadena de valor de la Industria, 2018. https://www.minsait.com/es/actualidad/insights/trazabilidad-en-la-cadena-de-valor-de-la-industria (accessed December 19, 2019)
[25] Radziwill, N. (2018). Blockchain revolution: How the technology behind Bitcoin is changing money, business, and the world. The Quality Management Journal, 25(1), 64–65.
[26] Luu, L., Chu, D. H., Olickel, H., Saxena, P., &Hobor, A. (2016, October). Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 254–269). ACM.
[27] Rojas, E. Qué son loscontratosinteligentes o "smart contracts"? (2019). Guíacompleta. https://es.cointelegraph.com/explained/what-is-a-smart-contract (accessed December 19, 2019)
[28] A. Ramos, R. Wert. (2014). Design thinking enespañol, http://www.designthinking.es/inicio/ (accessed December 19, 2019)
[29] O'REILLY RADAR, What is DevOps?. (2012). http://radar.oreilly.com/2012/06/what-is-devops.html (accessed December 19, 2019)
[30] DmitriySamovskiy's Blog, The Rise of DevOps. (2010). http://www.somic.org/2010/03/02/the-rise-of-devops/ (accessed December 19, 2019)
[31] John Willis, DevOps Culture (Part 1) – IT Revolution. IT Revolution. (2012). https://itrevolution.com/devops-culture-part-1/ (accessed December 19, 2019).
[32] ATLASSIAN, Gitflow Workflow |Atlassian Git Tutorial. (2017). https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (accessed December 19, 2019)
[33] Driessen, V. (2010). A successful Git branching model. URL http://nvie.com/posts/a-successful-git-branching-model. (accessed December 19, 2019)
[34] Gerber, A., & Craig, C. (2015). Introducing Git. In Learn Android Studio (pp. 145–187). Apress, Berkeley, CA.
[35] Janet Gregory, Lisa Crispin. Agile Testing Condensed: A Brief Introduction, ISBN-10:199922051X
[36] Pandit, P., & Tahiliani, S. (2015). AgileUAT: A framework for user acceptance testing based on user stories and acceptance criteria. International Journal of Computer Applications, 120(10).
[37] Mittal, S., & Martín, J. L. R. (2018). Netcentric system of systems engineering with DEVS unified process. CRC Press.
[38] Olivero, Miguel Angel, et al. "Security assessment of systems of systems." 2019 IEEE/ACM 7th International Workshop on Software

Engineering for Systems-of-Systems (SESoS) and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES). IEEE, 2019

[39] Gotel, O. C., & Finkelstein, C. W. (1994, April). An analysis of the requirements traceability problem. In Proceedings of IEEE International Conference on Requirements Engineering (pp. 94–101). IEEE.

[40] Ramesh, B., &Jarke, M. (2001). Toward reference models for requirements traceability. IEEE transactions on software engineering, 27(1), 58–93.

[41] Pohl, K. (2010). Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated.

[42] GraciaBandrés, M.A., GraciaMurugarren, J., Romero San Martín, D. (2015) TecsMedia: Metodologías de diseñocentradasenusuarios

[43] Knapp, J. (2012). The Design Sprint.

[44] Banfield, R., Lombardo, C. T., & Wax, T. (2015). Design sprint: A practical guidebook for building great digital products. "O'Reilly Media, Inc.".

[45] Olivero, Miguel Angel; Morales-Trujillo, L; Domínguez-Mayo, F. J.; Mejías, M., Systematic Development of ERP Modules using a Model-Driven Strategy Focusing on the Users,4th International Special Session on Advances Practices in Model-Driven Web Engineering in the 15th International Conference on Web Information Systems and Technologies, 2019

[46] Agile Manifesto. https://agilemanifesto.org/. (accessed December 19, 2019)

[47] IMF Business School, Metodologíaságiles de desarrollo. https://blogs.imf-formacion.com/blog/tecnologia/metodologias-agiles-de-desarrollo-201801/ (accessed December 19, 2019)

[48] Schaefer, A., Reichenbach, M., & Fey, D. (2013). Continuous integration and automation for DevOps. In IAENG Transactions on Engineering Technologies (pp. 345–358). Springer, Dordrecht.

[49] De Bayser, M., Azevedo, L. G., &Cerqueira, R. (2015, May). ResearchOps: The case for DevOps in scientific applications. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM) (pp. 1398–1404). IEEE.

[50] Erich, F. M. A., Amrit, C., & Daneva, M. (2017). A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 29(6), e1885.

[51] Virmani, M. (2015, May). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In Fifth International Conference on the Innovative Computing Technology (INTECH 2015) (pp. 78–82). IEEE.

Biographies

Leticia Morales Trujillo has a Bachelor's Degree in Health Engineering with a mention in Biomedical Engineering from the University of Seville since 2016 and a Master's Degree in Software and Technology Engineering from the University of Seville since 2018. She is currently a PhD student. Since 2016 researcher associated with the research group of Web Engineering and Early Testing (IWT2), belonging to the Department of Languages and Computer Systems of the University of Seville. She is currently enrolled in the 2018–2019 doctoral program at the University of Seville. At the researcher level, he has participated in several research projects at the national level. Among his most important research results are several contributions to national and international conferences and publications in JCR journals.

Miguel Ángel Olivero González is PhD candidate in Computer Science at the University of Seville. He has participated in various projects as a

researcher as a member in the Web Engineering and Early Testing Group (IWT2). He has been part of the organizing committee of different international conferences. He has made national and international stays and participated in both national and international projects. His current research interests are related to Model-Driven Engineering, Security, and the System of Systems context. Further information about his research activities and his list of publications can be found at https://investigacion.us.es/sisius/sis_showpub.php?idpers=25279

Francisco José Domínguez Mayo received the Ph.D. degree in computer science from the University of Seville, Seville, Spain, in July 2013. He is currently an associate professor with the Department of Computing Languages and Systems, University of Seville. He collaborates with public and private companies in software development quality and quality assurance. His lines of interesting research are plotted in the areas of continuous quality improvement and quality assurance on software products, and software development processes.

Julián Alberto García García was awarded his PhD in Computer Science by the University of Seville, Spain, in 2015. He is currently a Lecturer and Researcher with the Department of Computing Languages and Systems, University of Seville. Since 2008, he has participated in R&D projects as

a researcher in the Web Engineering and Early Testing Group (IWT2). His current research interests include the areas of business process management, business process modeling, Model-Driven Engineering and quality assurance. Julian is responsible for the BPM area and responsible for security in IWT2. He also participates as member of committee in several international conferences and journals.

Manuel Mejías Risoto obtained his Ph.D degree in Industrial Engineering at the University of Sevilla, Spain, in 1997. He has been several years teaching and researching in the field of Software Engineering. His current lines of research are plotted in the areas of methodological issues in software process, quality assurance and reference models in software production.