CENTRO DE CIENCIAS BÁSICAS

DOCTORADO EN

CIENCIAS APLICADAS Y TECNOLOGÍA

TESIS

AgileBPM Methodology - an Agile Development Methodology for Business Process Management Systems

PRESENTA

Roberto Dávila Campos

TUTORES

Dr. José Manuel Mora Tavarez

Dra. Paola Yuritzy Reyes Delgado

COMITÉ TUTORAL

Dra. Gabriela Citlalli López Torres

Dr. Jaime Muñoz Arteaga

Cd. Universitaria, junio, 2025

MTRO EN C. JORGE MARTÍN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO DE CIENCIAS BÁSICAS

P R E S E N T E

Por medio del presente como asesor designado del estudiante ***ROBERTO DÁVILA CAMPOS*** con ID **249394** quien realizó el **trabajo tesis** titulada ***AGILEBPM METHODOLOGY - AN AGILE DEVELOPMENT METHODOLOGY FOR BUSINESS PROCESS MANAGEMENT SYSTEMS***, un trabajo propio, innovador, relevante e inédito y con fundamento en la fracción IX del Artículo 43 del Reglamento General de Posgrados, doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que él pueda continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a *20* de *octubre* de *2025*.

Dr. José Manuel Mora Tavarez
Director de Tesis

c.c.p.- Interesado
c.c.p.- Coordinación del Programa de Posgrado

MTRO EN C. JORGE MARTÍN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO DE CIENCIAS BÁSICAS

PRESENTE

Por medio del presente como asesor designado del estudiante ***ROBERTO DÁVILA CAMPOS*** con ID **249394** quien realizó el trabajo de tesis titulada ***AGILEBPM METHODOLOGY - AN AGILE DEVELOPMENT METHODOLOGY FOR BUSINESS PROCESS MANAGEMENT SYSTEMS***, un trabajo propio, innovador, relevante e inédito y con fundamento en la fracción IX del Artículo 43 del Reglamento General de Posgrados, doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que él pueda continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

"Se Lumen Proferre"
Aguascalientes, Ags., a *20* de *octubre* de *2025*.

ATENTAMENTE

Dra. Paola Yuritzy Reyes Delgado
Co director de tesis

c.c.p.- Interesado
c.c.p.- Coordinación del Programa de Posgrado

MTRO EN C. JORGE MARTÍN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO DE CIENCIAS BÁSICAS

P R E S E N T E

Por medio del presente como asesor designado del estudiante ***ROBERTO DÁVILA CAMPOS*** con ID **249394** quien realizó la trabajo tesis titulada ***AGILEBPM METHODOLOGY - AN AGILE DEVELOPMENT METHODOLOGY FOR BUSINESS PROCESS MANAGEMENT SYSTEMS***, un trabajo propio, innovador, relevante e inédito y con fundamento en la fracción IX del Artículo 43 del Reglamento General de Posgrados, doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que él pueda continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a *20* de *octubre* de *2025*.

Dra. Gabriela Citlalli López Torres
Asesor de tesis

c.c.p.- Interesado
c.c.p.- Coordinación del Programa de Posgrado

CARTA DE VOTO APROBATORIO

MTRO EN C. JORGE MARTÍN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO DE CIENCIAS BÁSICAS

PRESENTE

Por medio del presente como asesor designado del estudiante ***ROBERTO DÁVILA CAMPOS*** con **249394** *numero* quien realizó el **trabajo tesis** titulada ***AGILEBPM METHODOLOGY - AN AGILE DEVELOPMENT METHODOLOGY FOR BUSINESS PROCESS MANAGEMENT SYSTEMS***, un trabajo propio, innovador, relevante e inédito y con fundamento en la fracción IX del Artículo 43 del Reglamento General de Posgrados, doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que él pueda continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"
Aguascalientes, Ags., a 23 de *octubre de 2025*.

Dr. Jaime Muñoz Arteaga
Asesor de tesis

c.c.p.- Interesado
c.c.p.- Coordinación del Programa de Posgrado

DICTAMEN DE LIBERACIÓN ACADÉMICA
PARA INICIAR LOS TRÁMITES DEL EXAMEN DE GRADO

Fecha de dictaminación (dd/mm/aaaa): 20/10/2025

NOMBRE: Roberto Olivio Campos **ID** 249394

PROGRAMA: Doctorado en Ciencias Aplicadas y Tecnología **LGAC (del posgrado):** Tecnologías de Ingeniería de Software y Objetos de Aprendizaje

MODALIDAD DEL PROYECTO DE GRADO: Tesis Tradicional (x) *Tesis por artículos científicos ( ) **Tesis por Patente ( ) Trabajo Práctico ( )

TITULO: AgileBPM Methodology - an Agile Development Methodology for Business Process Management Systems

IMPACTO SOCIAL (señalar el impacto logrado): Proporcionar una metodología de desarrollo agil y gratuito para proyectos de ciencia de datos, en pequeñas y medianas empresas de México.

INDICAR SEGÚN CORRESPONDA: **SI, NO, NA (No Aplica)**

	Elementos para la revisión académica del trabajo de tesis o trabajo práctico:
SI	El trabajo es congruente con las LGAC del programa de posgrado
SI	La problemática fue abordada desde un enfoque multidisciplinario
SI	Existe coherencia, continuidad y orden lógico del tema central con cada apartado
SI	Los resultados del trabajo dan respuesta a las preguntas de investigación o a la problemática que aborda
SI	Los resultados presentados en el trabajo son de gran relevancia científica, tecnológica o profesional según el área
SI	El trabajo demuestra más de una aportación original al conocimiento de su área
NO	Las aportaciones responden a los problemas prioritarios del país
SI	Generó transferencia del conocimiento o tecnológica
SI	Cumple con la ética para la investigación (reporte de la herramienta anti-plagio)
	El egresado cumple con lo siguiente:
SI	Cumple con lo señalado por el Reglamento General de Posgrados
SI	Cumple con los requisitos señalados en el plan de estudios (créditos curriculares, optativas, actividades complementarias, estancia, predoctoral, etc.)
SI	Cuenta con los votos aprobatorios del comité tutorial
N.A	Cuenta con la carta de satisfacción del Usuario (En caso de que corresponda)
SI	Coincide con el título y objetivo registrado
SI	Tiene congruencia con cuerpos académicos
SI	Tiene el CVU de la SECIHTI actualizado
SI	Tiene el o los artículos aceptados o publicados y cumple con los requisitos institucionales (en caso de que proceda)
	**En caso de Tesis por artículos científicos publicados (completar solo si le toca los por artículos)*
N.A	Aceptación o Publicación de los artículos en revistas indexadas de alto impacto según el nivel del programa
N.A	El (la) estudiante es el primer autor(a)
N.A	El (la) autor(a) de correspondencia es el Director (a) del Núcleo Académico
N.A	En los artículos se ven reflejados los objetivos de la tesis, ya que son producto de este trabajo de investigación
N.A	Los artículos integran los capítulos de la tesis y se presentan en el idioma en que fueron publicados
	***En caso de Tesis por Patente*
N.A	Cuenta con la evidencia de solicitud de patente en el Departamento de Investigación (anexarla al presente formato)

Con base en estos criterios, se autoriza continuar con los trámites de titulación y programación del examen de grado: SI X / No

FIRMAS

Elaboró:

*NOMBRE Y FIRMA DEL(LA) CONSEJERO(A) SEGÚN LA LGAC DE ADSCRIPCION: Dr. Francisco Javier Álvarez Rodríguez

* En caso de conflicto de intereses, designar un revisor diferente del PA de la LGAC correspondiente (revisor al director o referente del comité tutorial, designado por el Director)

NOMBRE Y FIRMA DEL COORDINADOR DE POSGRADO: Dr. Francisco Javier Álvarez Rodríguez

Revisó:

NOMBRE Y FIRMA DEL SECRETARIO DE INVESTIGACIÓN Y POSGRADO: Dr. Alejandro Padilla Díaz

Autorizó:

NOMBRE Y FIRMA DEL DECANO: Mtro. en C. Jorge Martín Alférez Chávez

Nota: procede el trámite para el Depto. de Apoyo al Posgrado

En cumplimiento con el Art. 24 fracción V del Reglamento General de Posgrado, que a la letra señala entre las funciones del Consejo Académico: Proponer criterios y mecanismos de selección, permanencia, egreso y titulación de estudiantes para asegurar la eficiencia terminal y la graduación y el Art. 28 fracción IX, anexar, archivar y dar el seguimiento correspondiente a este formato para su egreso hasta su titulación.

Elaborado por: D. Apoyo al Posg.
Revisado por: D. Control Escolar/D. Gestión de Calidad
Autorizado por: D. Control Escolar/ D. Apoyo al Posg.

Código: DO-DES-FO-15
Actualización: 02
Emisión: 12/08/21

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 February 2024, accepted 27 March 2024, date of publication 8 April 2024, date of current version 29 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3386167

RESEARCH ARTICLE

The Landscape of Rigorous and Agile Software Development Life Cycles (SDLCs) for BPMS: A Systematic Selective Literature Review

ROBERTO DÁVILA-CAMPOS[1], MANUEL MORA[1], PAOLA YURITZY REYES-DELGADO[1], JAIME MUÑOZ-ARTEAGA[1], AND GABRIELA CITLALLI LÓPEZ-TORRES[2]

[1]Basic Sciences Center, Autonomous University of Aguascalientes, Aguascalientes 20100, Mexico
[2]Economy and Management Sciences Center, Autonomous University of Aguascalientes, Aguascalientes 20100, Mexico

Corresponding author: Roberto Dávila-Campos (roberto.davila7@gmail.com)

ABSTRACT Business Process Management Systems (BPMS) are specialized Information Systems for the definition, execution, and management of business organizational processes including the participation of users, software, and data. To develop BMPS, several rigorous software development life cycles (SDLCs) have been reported in the literature in the last two decades. However, given the current organizational interest in agile SDLCs, we identified a knowledge gap on conceptual comparison between rigorous and agile SDLCs for BPMS. Consequently, this research aims to provide this missed conceptual review between both types of SDLCs. A systematic selective literature review (SSLR) research method upon 25 high-quality sources for the period 2000-2023 was conducted, and the conceptual review of eight SDLCs (four rigorous and four agile SDLCs) are reported using a rigorous and an agile SDLC pro form from core BPMS and agile paradigm literature. An empirical pilot evaluation on the usefulness, ease of use, and value of these conceptual reviews is also reported, and in overall the scores from BPMS practitioners and academics are satisfactory. On conceptual reviews, we found that three of the four rigorous SDLCs cover at most moderately the expected theoretical roles, phases-activities and work products, and none can be considered a de-facto standard to be used. Regarding the four agile SDLCs, we found that only two cover satisfactory the expected roles and phases-activities but partially work products. Nevertheless, the eight SDLCs are minimally documented, and consequently academics and practitioners lack full descriptions of them for their correct learning and practical utilization. Hence, our review provides theoretical and practical insights for discriminating both rigorous and agile SDLCs for BPMS and calls for further conceptual and empirical research to reach mature, standardized and systematic applied SDLCs for BPMS.

INDEX TERMS Business process management systems (BPMS), domain-agnostic process-aware information systems (PAIS), rigorous and agile BPMS SDLCs, usability metrics of usefulness, ease of use and value.

IEEE Access

Received 27 June 2025, accepted 23 July 2025, date of publication 1 August 2025, date of current version 18 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3594684

RESEARCH ARTICLE

Design and Evaluation of AgileBPM SDLC–An Agile SDLC for Business Process Management Systems

ROBERTO DAVILA-CAMPOS[1], MANUEL MORA[1], PAOLA YURITZY REYES DELGADO[1], SERGIO GALVÁN-CRUZ[1], AND GABRIELA CITLALLI LÓPEZ TORRES[2]

[1]Basic Sciences Center, Autonomous University of Aguascalientes, Aguascalientes 20100, Mexico

[2]Economy and Management Sciences Center, Autonomous University of Aguascalientes, Aguascalientes 20100, Mexico

Corresponding author: Roberto Davila-Campos (roberto.davila7@gmail.com)

ABSTRACT Business Process Management Systems (BPMS) are critical for enabling organizations to automate, monitor, and optimize their core processes. While agile software development approaches have transformed traditional systems engineering, their systematic application to BPMS development remains limited and underexplored. Existing agile SDLCs for BPMS are often incomplete, lack standardization, and provide insufficient documentation, hindering their adoption in both academic and professional contexts. This study addresses this gap by designing and validating AgileBPM SDLC, a novel, fine-grained Agile Software Development Life Cycle tailored specifically for BPMS. Grounded in Design Science Research Methodology (DSRM), the proposed model integrates best practices from the Scrum-XP framework and insights from four existing agile BPMS methodologies. The resulting AgileBPM SDLC includes three agile roles, four phases, fifteen activities, and twelve structured artifacts, all comprehensively documented through an open-access Electronic Process Guide (EPG). Validation involved expert review by senior evaluators and comparative assessment using a Rigor-Agility Framework. Results demonstrate the SDLC's theoretical soundness, alignment with agile principles, and practical applicability. By providing a detailed, reusable framework for agile BPMS development, this study makes a significant contribution to researchers, developers, and organizations seeking faster, more flexible, and well-documented approaches to process-aware information systems.

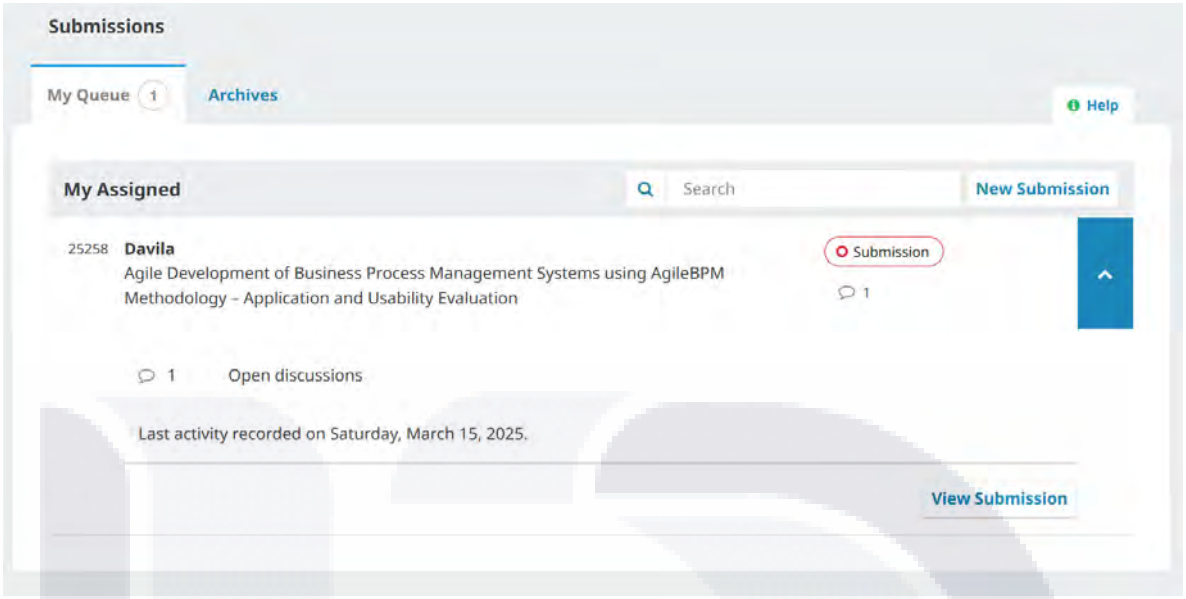
INDEX TERMS Agile SDLCs for BPMS, AgileBPM SDLC—an Agile SDLC for BPMS, business process management systems (BPMS), software development life cycles (SDLCs).

I. INTRODUCTION

A. CONTEXT AND MOTIVATION OF RESEARCH

Business Process Management (BPM) is a well-established field in organizational research [1], [2], [3], encompassing scientific and practical knowledge about analyzing, designing, implementing, monitoring, and improving business processes [4]. BPM focuses on frameworks, methodologies, and tools to enhance both the effectiveness (such as improving execution times) of business processes [5], [6]. As a core capability for modern businesses, BPM helps organizations achieve better outcomes and meet business metrics. It is thus considered crucial for maintaining competitiveness in today's business environment [7].

The foundation of BPM research revolves around the concept of a business process, which is defined as a sequence of actions — activities, events, and decision points —



AGRADECIMIENTOS

Quisiera primero que nada agradecer a Dios por darme la oportunidad de poder estar hoy aquí y haber logrado mis metas hasta ahora.

Me gustaría agradecer a todos mis profesores por compartir su tiempo y sus conocimientos que serán muy útiles a lo largo de mi vida.

A mis tutores y cotutores por su apoyo y su dedicación para poder completar con éxito este doctorado y la realización de los tres artículos científicos.

A mi amigo el Dr. Isaac Medina Sánchez por su apoyo durante la elaboración de este trabajo.

Por último, quiero agradecer a mi familia y a mi novia por el apoyo brindado durante este tiempo y darme la oportunidad de realizar mis estudios.

Gracias a todos por su valioso apoyo.

INDEX

INDEX OF TABLES

INDEX OF FIGURES

ABSTRACT IN SPANISH

Los Sistemas de Gestión de Procesos de Negocio (BPMS) son Sistemas de Información especializados para la definición, ejecución y gestión de procesos organizacionales de negocio, integrando la interacción entre software y personas (tanto usuarios como gerentes). Para desarrollar BPMS, se han reportado varios Ciclos de Vida de Desarrollo de Software (SDLCs) rigurosos en la literatura, y recientemente también se han reportado SDLCs ágiles iniciales. Sin embargo, a pesar del alto interés teórico y práctico en el desarrollo de BPMS desde un Enfoque Ágil, se ha identificado que los SDLCs ágiles iniciales para BPMS son incompletos en cuanto a los roles ágiles esperados, actividades y/o artefactos, y están mínimamente documentados. Como consecuencia, académicos y profesionales carecen de descripciones completas de ellos para su correcto aprendizaje y utilización práctica. En esta investigación, abordamos esta brecha de investigación y reportamos el diseño, la descripción completa de roles, actividades y artefactos, y la validación conceptual inicial del SDLC AgileBPM - un SDLC Ágil para BPMS– que fue elaborado utilizando una Metodología de Investigación en Ciencia del Diseño (DSRM). Los resultados iniciales de validación son satisfactorios, además, la investigación empírica también proporcionó resultados muy satisfactorios, finalmente se creó un caso de demostración con el nuevo SDLC AgileBPM con todos los procesos reportados.

ABSTRACT IN ENGLISH

Business Process Management Systems (BPMS) are specialized Information Systems for the definition, execution, and management of business organizational processes, integrating the interaction between software and people (both users and managers). To develop BMPS, several rigorous Software Development Life Cycles (SDLCs) have been reported in the literature, and recently, initial agile SDLCs have also been reported. However, despite the high theoretical and practical interest in BPMS development from an Agile Approach, it has been identified that the initial agile SDLCs for BPMS are incomplete regarding the expected agile roles, activities, and/or artifacts and are minimally documented. Consequently, academics and practitioners lack full descriptions of them for their correct learning and practical utilization. In this research, we address this research gap and report the design, the full description of roles, activities, and artifacts, and initial conceptual validation of AgileBPM SDLC - an Agile SDLC for BPMS– which was elaborated using a Design Science Research Methodology (DSRM). Initial validation results are satisfactory, further the empirical research also provided very satisfactory results, finally a demo case was created with the new AgileBPM SDLC with all the processes reported.

1 INTRODUCTION

1.1 CONTEXT OF THE RESEARCH PROBLEM

Software Systems have become essential in any business and work of millions of people around the world. Nowadays, most companies in the world are using a Software System or are developing one that expects to help in their Business Process Management. Most of these Software Systems, are currently, based on Web platforms, and they are well-known as Web Applications that bring all the benefits from a Software System with the web capabilities such as available anywhere, not installation required and run on any compatible device with a Web browser (Navarro, 2009).

Business worldwide organizations are always searching for developing new useful Software Systems with positive features such as ease of use, secure, and valuable. However, these expectations are not easily achieved due to the long time that takes the development of these software systems, the rework by wrong requirements, and off-budget events. Nowadays fast software system development approaches play a key role in any industry, and any significant delay could affect customer satisfaction or break any contractual agreement (Hughes et al., 2017).

To accelerate the software development, in the Software Engineering discipline, emerged the Agile Methodologies that aim to improve the speed of the application development with all desirable features like quality, and ease of use (Petersen & Wohlin, 2009).

The main benefits that Agile Methodologies bring to companies and developers (Shankarmani et al., 2012) are:

- Created just needed documentation.
- Focus more on the application.
- Iterative development brings helpful feedback from stakeholders.
- Low rework amount.
- Transparency brings real-time updates on the status of development.

Developing fast software is also possible using Low-Code Business Process Management (BPM) platforms. The expected aim of these development tools consists of the Software System analyst can implement quickly and easily a simple but useful Web Software System without too much programming knowledge and in a shortened period. BPMS are defined *"**as a (suite of) software application(s) that enable the modeling, execution, technical and operational monitoring,***

and user representation of business processes and rules, based on the integration of both existing and new information systems functionality that is orchestrated and integrated via services" (Ravesteyn & Batenburg, 2010a, p. 496). Working jointly with Agile Methodologies and BPMS could bring to organizations the fast software development that current business needs are demanding.

However, the main problem is that agile practitioners prefer regular software development using traditional programming languages like Java, JavaScript, PHP, Phyton, and others (Barabino et al., 2014).

The utilization of these non-agile development tools, thus, can affect the schedule and the budget of the projects. Additionally, there are a few studies related to Agile Methodologies for BPMS so that the developers who are trying to work with Low-Code BPMS platforms encounter many developing problems. Working without any methodology on BPMS could cause low-quality software due to the lack of a guided development process.

We believe that an Agile Methodology combined with a Low-Code platform on BPMS could impact positively the development of Web Software Systems with the expected attributes of quality, security, and ease of use that fit with the project schedule and budget.

1.2 MOTIVATION AND RELEVANCE OF THE RESEARCH PROBLEM

Several global business studies report that the utilization of agile development methodologies is a frequent practice in large-, medium- and small-sized organizations (Hoda et al., 2018). Similarly, the market for Low-Code development BPMS platforms will grow in the next 5 years (2020-2025) (Markets and Markets, 2020). Another professional website reports "***The global low-code application development platform market size was valued at USD 11.45 billion in 2019 and is expected to grow at a compound annual growth rate (CAGR) of 22.7% from 2020 to 2027. Increasing digital transformation in the IT and telecom industry, increased responsiveness to the business, and rising need for customization and scalability are the major factors driving the market growth.***" (Grand View Research, 2020).

Figure 1 shows the Grand View Research Inc. forecast for low-code platforms with historical data.

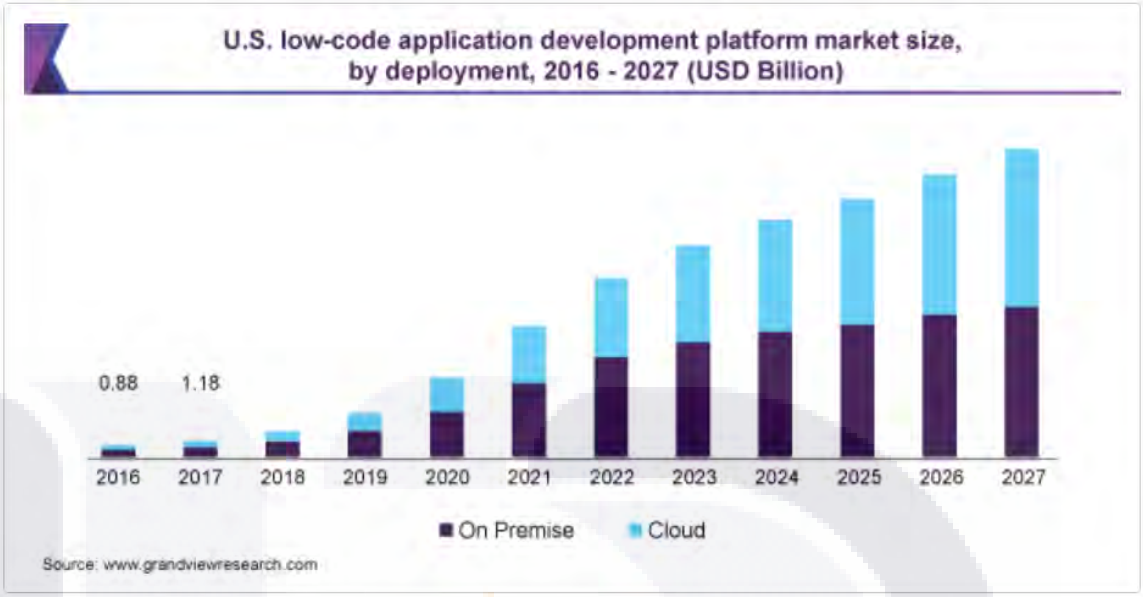


Figure 1 - Low-Code applications market size forecast by Grand View Research, Inc.

The Forrester Wave research (Rymer, 2017) found that low-code platforms were growing and after a survey of software development leaders that were using those platforms, they found three key features of this kind of platform:

- Speed up application and innovation delivery.
- Prove useful for large-scale applications.
- Contribute to AD&D's (application development and delivery) move to public clouds.

With these two technological trends, and the current need for multiple Web Information Software Systems in the organizations for help in their Business Digitalization process (Petersen & Wohlin, 2009), the business organizations require agile software development methodologies that can produce useful, easy use, secure and valuable product software (i.e. to fit the product quality), as well as they need that these agile software development methodologies help to fit the project schedule and budget.

1.3 FORMULATION OF THE RESEARCH PROBLEM

1.3.1 RESEARCH PROBLEM

Consequently, based on the previous research context described, we can identify the research problem directly as "**the lack of development methodologies for Web Software Systems -of type BPMS- that be considered by the software developers as agile, ease of use, useful, compatible, and valuable**".

1.3.2 RESEARCH QUESTIONS AND HYPOTHESES

RQ.1 What is the state of the art – contributions and limitations- on agile and non-agile development methodologies for Business Process Management systems?

H0.1 There is no need for an agile development methodology for Business Process Management systems

RQ.2 What is the state of the art – capabilities, and limitations – of open-source low-code Business Process Management development platforms?

H0.2 There are no powerful open-source low-code Business Process Management development platforms.

RQ.3 What elements of Agile Development and Business Process Management System Development Methodologies can be used to elaborate an Agile Business Process Management System Development Methodology that can be evaluated theoretically valid from a Panel of Experts?

H0.3 There are no elements of Agile Development and Business Process Management System Development Methodologies that can be used to elaborate an Agile Business Process Management System Development Methodology that can be evaluated theoretically valid by a Panel of Experts.

RQ.4 Can the new elaborated Agile Business Process Management System Development Methodology be documented in an Electronic Process Guide (EPG)

and be evaluated as agile, useful, easy to use, compatible, and valuable by a pilot group of Software Engineering academics and practitioners?

H0.4.1 The newly elaborated Agile Business Process Management System Development Methodology cannot be documented in an Electronic Process Guide (EPG).

H0.4.2 The newly elaborated Agile Business Process Management System Development Methodology is not considered agile, useful, easy to use, compatible, and valuable by a pilot group of Software Engineering academics and practitioners.

1.3.3 GENERAL AND SPECIFIC RESEARCH OBJECTIVES

To design conceptually a development methodology for Business Process Management systems, and document it in an Electronic Process Guide, that be evaluated as agile, useful, easy to use, compatible, and valuable for a pilot group of Software Engineering academics and practitioners.

1.3.4 CONTRIBUTIONS AND DELIVERABLES OF THE RESEARCH

In this research proposal, it is expected to produce the following products:

For the Software Engineering Theory:

- One research paper for an indexed journal with the theoretical analysis on “The State of the Art on Open-Source Business Process Management Low-Code Platforms”
- One research paper for an indexed journal with the theoretical analysis on “The State of the Art on Development Methodologies for Business Process Management Systems”
- One submitted research paper for an indexed journal with the theoretical analysis and empirical evaluation of the AgileBPM Methodology – an agile Methodology for BPM Systems

For the Software Engineering Practice:

- A new AgileBPM Methodology – an agile Methodology for BPM Systems, available in a web-based free-cost access EPG (Electronic Process Guideline)
- A new Ph.D. graduate in the Software Engineering area

1.4 GENERAL DESCRIPTION OF THE RESEARCH METHODOLOGY

In this research, it is proposed to use a Design Science Research approach (van Brocke et al., 2020; Peffers et al., 2007). “Design Science Research (DSR) is a problem-solving paradigm that seeks to enhance technology and science knowledge bases via the creation of innovative artifacts that solve problems and improve the environment in which they are instantiated. The results of DSR include both the newly designed artifacts - represented by constructs, and/or models, and/or methods, and/or instantiations -, and design knowledge (DK)”.

1.4.1 OVERVIEW OF THE RESEARCH METHODOLOGY

The specific DSR methodology to conduct is the Design Science Research Methodology proposed by Peffers *et al*. (2007a). It has six activities as follows:

- **Activity 1**: Problem identification and motivation. “***Define the specific research problem and justify the value of a solution. Justifying the value of a solution accomplishes two things: it motivates the researcher and the audience of the research to pursue the solution and to accept the results and it helps to understand the reasoning associated with the researcher’s understanding of the problem***”.
- **Activity 2.1**: Define the objectives for a solution. “***Infer the objectives of a solution from the problem definition and knowledge of what is possible and feasible. The objectives can be quantitative, such as terms in which a desirable solution would be better than current ones, or qualitative, such as a description of how a new artifact is expected to support solutions to problems not hitherto addressed***”.
- **Activity 2.2**: Review the State of the Art. Review the state of the art on the main element to be designed and identify the main contributions and limitations.
- **Activity 3**: Design and development. *“**Create the artifact. Such artifacts are potentially constructing, models, methods, or instantiations (each defined broadly). Conceptually, a design research artifact can be any designed object in which a research contribution is embedded in the design. This activity includes determining the artifact’s desired functionality and its architecture and then creating the actual artifact**”.*

- **Activity 4**: Demonstration. *"**Demonstrate the use of the artifact to solve one or more instances of the problem. This could involve its use in experimentation, simulation, case study, proof, or other appropriate activity**".*
- **Activity 5**: Evaluation. *"**Observe and measure how well the artifact supports a solution to the problem. This activity involves comparing the objectives of a solution to actual observed results from use of the artifact in the demonstration. At the end of this activity the researchers can decide whether to iterate back to activity 3 to try to improve the effectiveness of the artifact or to continue to communication and leave further improvement to subsequent projects**"*. The specific Evaluation methods to be used will be:
  - Evaluation Conceptual from a Panel of Experts.
  - Evaluation from a Proof of Concept.
  - Empirical survey-based evaluation from a pilot sample of Software Engineering professionals.
- **Activity 6**: Communication. "***Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences such as practicing professionals, when appropriate***".

1.4.2 TIMELINE – SEMESTERS, ACTIVITIES, AND DELIVERABLES

Table 1 displays the timeline and schedule expected to work in the activities described.

Table 1 - Activity schedule.

Phases	2021	2022	2023	2024
Activities 1 and 2.1 a) Background and history of the problem. b) Problematic situation. c) Type and purpose of research. d) Relevance. e) Objectives, questions, and hypotheses/research propositions.	x			
Activity 2.2 Review the State of the Art a) Theories bases. b) Studies related. c) Contributions and limitations of related studies.	x	x		
Activity 3 Design and Development of Artifact a) Application or creative-deductive relational conceptual design model.			x	x
Activities 4 and 5 – Demonstration and Evaluation a) Validation of content by a panel of experts. b) Validation by logical argument. c) Validation for proof of concept of the artifact.			x	x
Activities 6 – Communication a) Write and submit research paper 1. b) Write and submit research paper 2. c) Write and submit research paper 3.			x	x

2 RESEARCH METHODOLOGY

This Ph.D. research uses an adapted Design Research Methodology from two core studies on Design Science Research (Hevner et al., 2004) (Peffers et al., 2007) complemented with additional research steps: Selective Systematic Literature Review method (Cooper 1988), Conceptual Design (Mora, 2009) Conceptual Validation from Panel of Experts (Beecham et al. 2005), Empirical Validation with Statistical Analysis (Wohlin et al., 2012), and Guide for Scientific Reports in Software Engineering (Shaw, 2003). Figure 2 displays the steps for DSR.

Table 2 summarizes steps, purpose, complementary research methods, and expected outcomes.

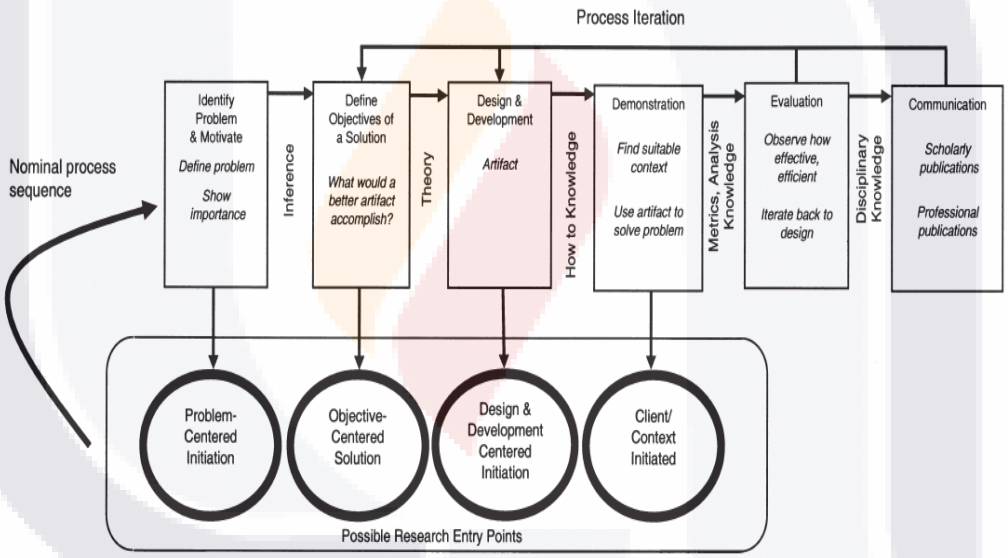


Figure 2 - DSRM for information systems.

Table 2 - Design Science Research Methodology (DSRM) with complementary research methods

DSRM Steps	Purpose	Complementary research methods	Outcomes
Step 1) Design problem identification and motivation.	To state the expected overall research goal that delimits the scope of the research, the research questions that focus on the knowledge gaps of interest, and the motivations to pursue the research design. (For these aims is required to conduct a Review of the State of the Art on the specific problem.).	• Conceptual Literature Review (CLR), or • Systematic Literature Review (SLR), or • Selective Systematic Literature Review (SSLR).	• Research overall goal statement. • Research questions. • Research motivation statements. • Review of the State of the Art.
Step 2) Definition of the design objectives and restrictions for the expected artifact.	To define the specific design objectives (i.e. expected qualities in the designed artifact), design restrictions (i.e. the limitations on time, cost and resources utilized to design the artifact), design approach (i.e. analytics, axiomatic or heuristic), design theoretical sources (i.e. the design materials), and design components (i.e. the specific design building-blocks).	• Conceptual Design.	• Design problem identification and motivation. • Definition of the Design Objectives, Design Restrictions, Resign Approach, Design Theoretical Sources, and Design Components for the expected Artifact.
Step 3) Design and development of the artifact.	To design and implement the expected artifact guided-controlled by the design objectives and restrictions, and using the agreed design approach, design theoretical sources and design components.	• Heuristic Design.	• Conceptual designed artifact. • Implemented designed artifact.
Step 4) Demonstration of the artifact (Proof of Concept).	To demonstrate the designed and implemented artifact and conduct initial verification.	• Verification by a Panel of Experts	• Conceptual Verification by a Panel of Experts.
Step 5) Evaluation of the artifact.	To conduct empirical evaluation of the designed and implemented artifact.	• Survey or Experimental Methods.	• Empirical Validation with Statistical Analysis.
Step 6) Communication of research results.	To generate a structured scientific report (i.e. Thesis, Technical Report, Chapter, Conference Proceeding document, or Journal article) of results and communicate them in academic outlets.	• Scientifc writing guidelines.	• Structured Scientific Report.

2.1 MAIN ACTIVITIES

For Activities 1 and 2.1 the following actions will be implemented:

- Background and history of the problem.
- Problematic situation.
- Type and purpose of research.
- Relevance.
- Objectives, questions, and hypotheses/research propositions.

For Activity 2.2 the following actions will be implemented:

- Theories bases.

- Studies related.
- Contributions and limitations of related studies.

For Activity 3 the following actions will be implemented:

- Application or creative-deductive relational conceptual design model.

For Activities 4 and 5 the following actions will be implemented:

- Validation of content by a panel of experts.
- Validation by logical argument.
- Validation for proof of concept of the artifact.

For Activities 6 the following actions will be implemented:

- Write and submit research paper 1.
- Write and submit research paper 2.
- Write and submit research paper 3.

For more details about the Activities please check Table 1-1.

2.2 OBJECT AND SUBJECTS OF STUDY

This Ph.D. dissertation has the following objects of study:

- Scrum – Agile development framework.
- eXtreme Programming (XP) – Agile development.
- BPMS – Business Process Management Systems.

The subjects of study are:

- Practitioners and academics are interested in agile BPMS development methods.
- Pilot sample software.

Agile methodologies, Business Process Management Systems are the based for this research, a pilot sample software is going to be developed, and will be evaluating with practitioners and academics interested in the BPMS topic.

2.3 MATERIALS AND EQUIPMENT

For this work we are going to use the following materials and equipment:

- Articles of research, chapters, conference papers, and book related to the topics of Software Engineering, Software Development, Agile Methodologies, BPM, and BPMS.
- VM Server
- Laptop / PC
- Open-source tools:
  - ProcessEdit
  - Joget
  - Visual Studio Code

2.4 RESEARCH EVALUATION METHODS

"***The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods***." (Hevner & Ram, 2004, p. 85). "***IT artifacts can be evaluated in terms of functionality, completeness, consistency, accuracy, performance, reliability, usability, fit with the organization, and other relevant quality attributes. When analytical metrics are appropriate, designed artifacts may be mathematically evaluated."*** (Hevner & Ram, 2004, p. 85). Table 3 shows different evaluation methods for the Design Research created by Hevner (2004). An experimental evaluation method is selected to evaluate the new AgileBPM Methodology.

Table 3 - Design Research Evaluation Methods.

Design Evaluation Methods	
1. Observational	Case Study: Study artifact in depth in business environment.
	Field Study: Monitor use of artifact in multiple projects.
2. Analytical	Static Analysis: Examine structure of artifact for static qualities (e.g., complexity)
	Architecture Analysis: Study fit of artifact into technical IS architecture
	Optimization: Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior
	Dynamic Analysis: Study artifact in use for dynamic qualities (e.g., performance)
3. Experimental	Controlled Experiment: Study artifact in controlled environment for qualities (e.g., usability)
	Simulation - Execute artifact with artificial data

Design Evaluation Methods	
4. Testing	Functional (Black Box) Testing: Execute artifact interfaces to discover failures and identify defects
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility
	Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility

Based on methodological recommendations we are going to apply the specific techniques:

- Validation of Content by a Panel of Experts.
- Validation by Proof of Concept of Designed Artifact.
- Empirical Validation by a Pilot Survey Study / Demo Case Scenario (with an international sample of software academicians and practitioners).

2.5 RESTRICTIONS AND LIMITATIONS

Time will be the biggest limitation for this work, there are only 3 or 4 years available to finish the project. The budget will be also a limitation for this Ph.D. study.

The scope for this AgileBPM Methodology is for micro and small projects with five to ten people and three to six months within $10,000 to USD 20,000 of budget.

3 THEORETICAL BACKGROUND

3.1 THEORETICAL FOUNDATIONS

3.1.1 ON SOFTWARE ENGINEERING

Software is the key element for this research and the root of the Software Engineering discipline, software is defined by IEEE (2021) as "***computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system***", Pressman & Maxim (2015) defines computer software as "***the product that software professionals build and then support over the long term. It encompasses programs that execute within a computer of any size and architecture, content that is presented as the computer programs execute, and descriptive information in both hard copy and virtual forms that encompass virtually any electronic media***".

Software Engineering is a branch of Computer Science that splits into twelve different areas as Algorithms & Data Structures, Programming Languages, Architecture Operating Systems and Networks, Software Engineering, Databases & Information Retrieval, Artificial Intelligence & Robotics, Graphics, Human-Computer Interaction, Computational Science, Organizational Informatics, and Bioinformatics (Denning, 1999). In this Ph.D. dissertation, we will focus only on Software Engineering which has all the foundations that our research needs to be done.

There are many **Software Engineering** definitions provided by different authors. For instance, S.W. Humphrey stated (1988, p. 82) that Software Engineering "***refers to the disciplined application of engineering, scientific, and mathematical principles and methods to the economical production of quality software.".*** Abran and Moore (2014, p. xxxi) defined **Software Engineering** as ***"the application of a systematic, disciplined, quantifiable, approach to the development, operation, and maintenance of software; that is, the application of engineering to software".*** Finally, for Pressman and Maxim (2015, p. 14) Software Engineering "***encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.***" IEEE (2021) states that Software Engineering is a "***systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software***".

SWEBOK (Abran & Moore, 2014) divides **Software Engineering** into fifteen Knowledge Areas (KA) that are: Software Requirements, Software Design Software Construction, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, **Software Engineering processes**, Software Engineering Models and Methods, Software Quality, Software Engineering Professional Practice, Software Engineering Economics, Computing Foundations, Mathematical Foundations, and Engineering Foundations.

In this Ph.D. dissertation, we need to focus on **Software Engineering Processes** KA which is defined such "***software engineering processes are concerned with work activities accomplished by software engineers to develop, maintain, and operate software, such as require meets, design, construction, testing, configuration management, and other software engineering processes."*** (Abran & Moore, 2014, pp. 8–1).

Software Engineering Process is divided into five areas (see Figure. 3) described by the SWEBOK as Software Process Definition, Software Life Cycles, Software Process Assessment and Improvement, Software Measurement, and Software Engineering Process Tools. Every process has also its subprocesses.

Software Process Definition is where all the processes are defined, every process has an input and output, and the decomposition of the work activities. Software Life Cycles is where the software requirements are transformed into deliverable products, we will talk more about this area below. The software Process Assessment and Improvements area is meant to evaluate the software processes and improve every cycle implementing the Plan-Do-Check-Act model. Software Measurement is the area where the baselines are created before implementing a new process to know what process is providing betters results (Abran & Moore, 2014).

Oktaba and González defined **Software Process** as "***a composition of phases, activities, artifacts, and resources (including the humans)***" (1998, p. 229). Every single process needs a set of tools and resources to be accomplished, humans are part of those resources, and they must correctly manage the activities.

The software Life Cycle area is our focus in this Ph.D. dissertation, "***a software development life cycle (SDLC) includes the software processes used to specify and transform software requirements into a deliverable software product. A software product life cycle (SPLC) includes a software development life cycle plus additional software processes that provide for deployment, maintenance, support, evolution, retirement, and all other inception to retirement processes for a software product.***" (Abran & Moore, 2014, p. 8–4). In this area the relationship and temporal ordering from the processes are defined, some processes may be run at the same time to provide a

shared output while other processes must wait for that output so that they can start working.

Categories of **Software Processes** defined four categories: Primary processes are for the development, operation, and maintenance of software. Supporting processes support primary processes when needed like configuration management, quality assurance, and verification and validation. Organizational processes support the software engineering inside an organization and include training, process measurement analysis, infrastructure management, portfolio, and reuse management, organizational process improvement, and management of software life cycle models. The cross-project process works on two or more projects; reuse, software product line, and domain engineering are part of this category. Project management processes include planning and estimating, resource management, measuring and controlling, leading, managing risk, managing stakeholders, and coordinating the primary, supporting, organizational, and cross-project processes of software development and maintenance projects. Depending on the organization it could also be more processes to be developed to cover all the needs like process activities focusing on software quality (Abran & Moore, 2014).

To have a better understanding of Software processes from SWEBOK (Abran & Moore, 2014), it is possible to see similarities with the four categories from CMMI Project Management, Engineering, Support, and Process Management (Capability Maturity Model Integration) (*CMMI for Development, Version 1.3*, n.d.). Table 4 shows the CMMI categories and their process with Software Processes defined in SWEBOOK (Abran & Moore, 2014).

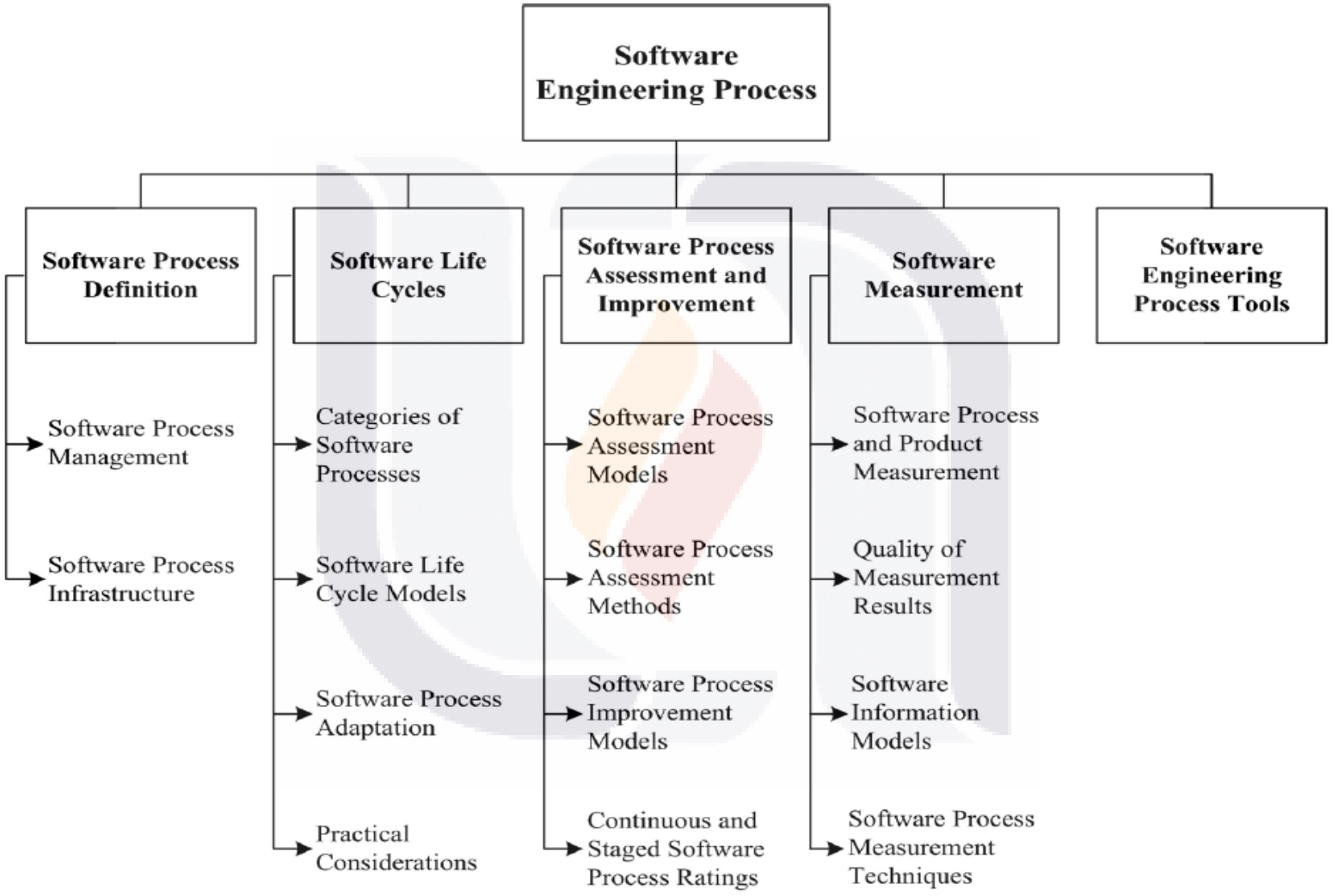


Figure 3 - Software Engineering Process breakdown.

Table 4 - CMMI Categories and their processes with Process Software categories.

CMMI Category	CMMI Process Area	Software Process Category
Project Management	Integrated Project Management (IPM) Project Monitoring and Control (PMC) Project Planning (PP) Quantitative Project Management (QPM) Requirements Management (REQM) Risk Management (RSKM) Supplier Agreement Management (SAM)	Cross-project Process
Engineering	Product Integration (PI) Requirements Development (RD) Technical Solution (TS) Validation (VAL) Verification (VER)	Primary Processes
Support	Causal Analysis and Resolution (CAR) Configuration Management (CM) Decision Analysis and Resolution (DAR) Measurement and Analysis (MA) Process and Product Quality Assurance (PPQA)	Supporting Processes
Process Management	Organizational Process Definition (OPD) Organizational Process Focus (OPF) Organizational Process Performance (OPP) Organizational Performance Management (OPM) Organizational Training (OT)	Organizational Processes

The software allows a great variety of Software Life Cycles; linear models have different phases of software development that need to be completed sequentially, software requirements are rigorously controlled, and every change needs to supervise and authorized by Software Configuration Management KA. Agile SDLC defined the requirements as a high-level state and that requirements can be detailed or changed during the development to facilitate the evolution of the software (Abran & Moore, 2014).

Software Process Adaptation defines software development life cycles and the software product life cycles, and the individual process often needs to be adapted. Sometimes does not makes sense to implement all the process defined in the cycles due to business rules, culture, and size of the company. There are

situations where it is necessary to put more control on the processes, so they put more processes into the development cycles (Abran & Moore, 2014).

There should be a lot of Practical Considerations a lot of software processes should be recognized as idealizations that must be adapted to reflect the realities of software development within the organization and business context. Most of the time the software development cycles need to be adapted to every organization to have a better solution for the business (Abran & Moore, 2014).

The software has become a basic need almost in every human activity for that reason the software demand has increased year by year and is very important that software engineers can accomplish every software development in time, Software Engineering defines Tools, Methods, Processes, and Quality Tools (Pressman & Maxim, 2015): Tools can be automated or semi-automated that are integrated with the methods and provide support to them. Methods provide a how-to create the software step by step, every method has different phases, and every phase has some tasks that every role must implement. A Process is the set of activities, actions or tasks to be completed to create a product. It is important to say that a Process is not rigid, this means that the software engineers can select the appropriate activities, actions, or tasks that best fit into the developed product. The Quality Focus establish that all the Tools, Methods, and Process should be always implemented with the quality in mind to satisfy the stakeholders that sponsored the project.

Parnas (2010) stated that software development is lacking disciplined, most of the time software developers do not follow any rules, predefined steps, or methodologies or they use risky shortcuts in the development. All these errors produce sloppy software and can produce major problems for the users and companies.

As we have seen **Software Engineering** is very important in software development and must be implemented in every development and better practices need to be created in the future to improve current results. Garousi *et al.* (2020) studied the relevance of Software Engineering research after 50 years of SE. The authors found some root causes that made the research irrelevant and made some suggestions. Figure 4 created by Garousi *et al.* (2020) maps the root causes with the suggestions.

There have been a lot of SDLC processes through time, Rodríguez *et al*. (2009) compared and classified different processes with the key values of “specification rigor” and “agility” (see Figure. 5). The results showed that most of the methodologies have a medium value for both references. It is very important to mention that there was not an SDLC that fulfilled high agility with specification rigor. In this Ph.D. dissertation, we will try to fill the gap, or get the closest as possible, between both key values.

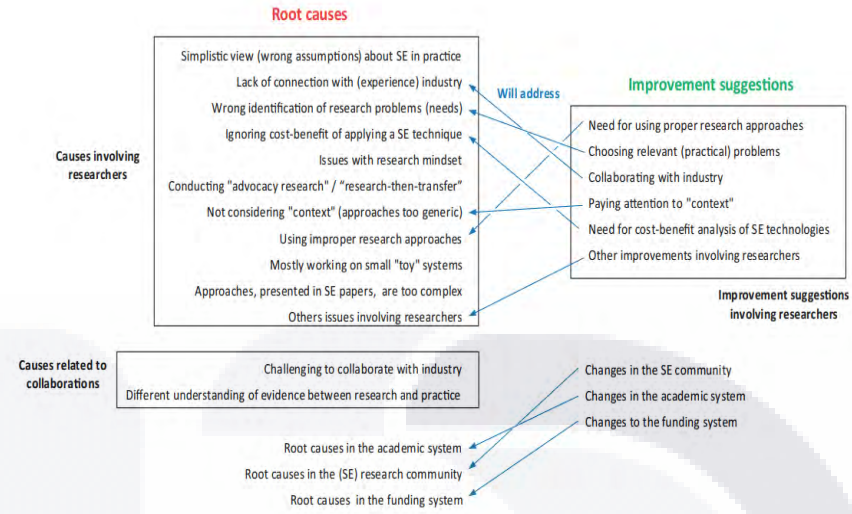


Figure 4 - Suggestions to improve software engineering research to make it relevant.

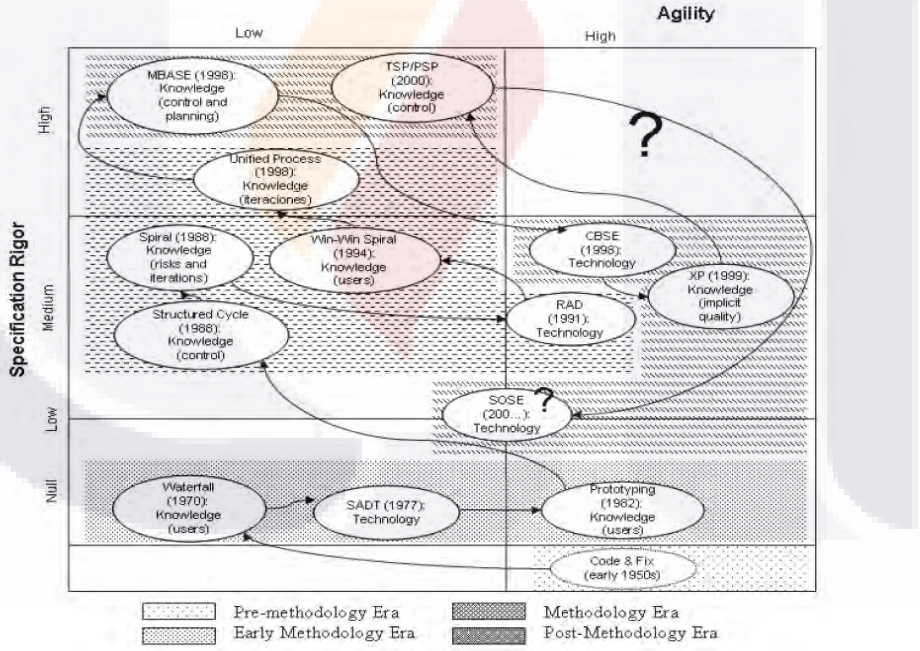


Figure 5 - SDLC evolution and comparison.

Hence, this section provides the following important concepts:

Software

"*Computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system.*" (*ISO/IEC/IEEE 24765:2017(En), Systems and Software Engineering — Vocabulary*, 2021)

"***Computer software is the product that software professionals build and then support over the long term. It encompasses programs that execute within a computer of any size and architecture, content that is presented as the computer programs execute, and descriptive information in both hard copy and virtual forms that encompass virtually any electronic media.***" (Pressman & Maxim, 2015, p. 1).

Software Engineering

"*Systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software.*" (*ISO/IEC/IEEE 24765:2017(En), Systems and Software Engineering — Vocabulary*, 2021)**.**

"***Encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software."*** (Pressman & Maxim, 2015, p. 14)

"***Encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.***" (Pressman & Maxim, 2015, p. 14).

Software Engineering Processes

"***Software engineering processes are concerned with work activities accomplished by software engineers to develop, maintain, and operate software, such as require meets, design, construction, testing, configuration management, and other software engineering processes.***" (Abran & Moore, 2014, pp. 8–1).

Software Process

"***A composition of phases, activities, artifacts, and resources (including the humans).***" (Oktaba & Ibargüengoitia González, 1998, p. 229)

Software Life Cycle

"***A software development life cycle (SDLC) includes the software processes used to specify and transform software requirements into a deliverable software product. A software product life cycle (SPLC) includes a software development life cycle plus additional software processes that provide for deployment, maintenance, support, evolution, retirement, and all***

other inception to retirement processes for a software product." (Abran & Moore, 2014, p. 8–4).

3.1.2 ON AGILE DEVELOPMENT PARADIGM

3.1.2.1 REVIEW OF FUNDAMENTAL CONCEPTS OF AGILITY

Over the last years, the **Agile Methodologies** have been chosen as high-speed methodologies for developing volatile internet applications, and web development (Paulk, 2002). Following the **Agile manifesto** (Beck et al., 2001) these methodologies focus on people, working software, customer collaboration, and responding to change in a very easy way. Following the four principles from the **Agile manifesto**, it is possible to work on the most important things, when software development is in progress, improves the development time, and keeps the work aligned with the company's budget.

Linear SDLC and Agile SDLC have a lot of differences, Hong *et al.* (2011) listed most of them (see Table 5) between those two software development cycles. **Agile SDLC** supports short development cycles and can be adapted very quickly to any change, instead linear **SDLC** have very long cycles so that any change could have a big impact on the development. IEEE (2021) defines agile development as "***software development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continuous stakeholder feedback***".

Both **SDLC** must have their rigid steps even though agile models are more flexible it is very important to follow the necessary steps to have the desire results. The short development cycles allow to the stakeholders know the direction of the project in almost real-time so that they can make any adjustment avoiding unnecessary rework and waste of time. A key difference is the management of the user requirements, it is not very common that stakeholders have all the requirements at the beginning of the project, they can cause a lot of problems in a linear **SDLC**, something that can be more manageable in an agile **SDLC**.

In the late 1990s, agile methods emerged and offered lightweight processes with a focus on people and interactions (Hoda et al., 2018). Nowadays the last State of Agile survey (*State of Agile Survey*, 2021) reported that 95% of the surveyed companies are applying agile methods within the organization. 18% of them have all their teams working with agile methods, and 33% of the companies have more than half of their teams working with agile. Finally, the report stated that 75% of the respondents are using **Scrum**. Hoda *et al*. (2018) expect that agile software development continues growing working together with the new technologies and trends like the Internet of Things, Big Data, Virtua-Reality, and more.

Table 5 - Main differences between linear and agile SDLC.

	Agile IS	**Non agile IS**
Applicable context	More fluid user requirements	Relatively stable user requirements.
Identification of user requirements	Users are constantly solicited for new requirements; emphasis on adaptivity to changing environments	User requirements are typically identified at the start of the development cycle, with emphasis on planning and predicting.
Number of development cycles	Many short development cycles	One long development cycle
Development steps within each development cycle	Rigid steps	Rigid steps
Functions available when system is first released.	System only provides a limited set of functions when first released	System is expected to deliver a full set of functions when first released
Goal in each development cycle	Each release has limited scope, i.e., each release delivers only a few valuable functions.	A major release that comes with a complete set of functions.
Typical release frequency	Frequent; typically, every few weeks to every few months.	Infrequent; typically, after a few years.
Example systems	iPhone apps, company intranets, Web-based systems, software as a service, etc.	Operational systems, enterprise resource planning, office automation systems, etc.

The Agile manifesto is based on twelve principles (Beck et al., 2001), every principle does emphasis different situations that make agile work as a discipline. Laanti *et al*. (2013) analyzed every principle from the agile manifesto as showed in Table 7.

Table 6 - Emphasis of every single agile principle on the manifesto.

Agile Principle	Emphasis
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Customer satisfaction, Continuous delivery, value, early deliveries
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Adaptability, competitiveness, customer benefit
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	Frequent deliveries
Business people and developers must work together daily throughout the project.	Collaboration
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Motivated individuals, good environment, support, trust
The most efficient and effective method of conveying information to and within a development team is face-toface conversation.	Efficiency, communication
Working software is the primary measure of progress.	Measure progress via deliverables
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Sustainability, people
Continuous attention to technical excellence and good design enhances agility.	Focus on technical excellence,
Simplicity – the art of maximizing the amount of work not done –is essential.	Simplicity, optimize work
The best architectures, requirements, and designs emerge from self-organizing teams.	Self-organization
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Built-in improvement of efficiency and behavior

Laanti *et al*. (2013) recollected some definitions of "**Agile**" found in literature, Ambler (2007) defined it as the "***iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams with "just enough" ceremony that produces high-quality software in a cost-effective and timely manner which meets the changing needs of its stakeholders".*** Schuh (2004) ***"Building software by empowering and trusting people. Acknowledging change as a norm, and promoting constant feedback. Producing more valuable functionality faster."***

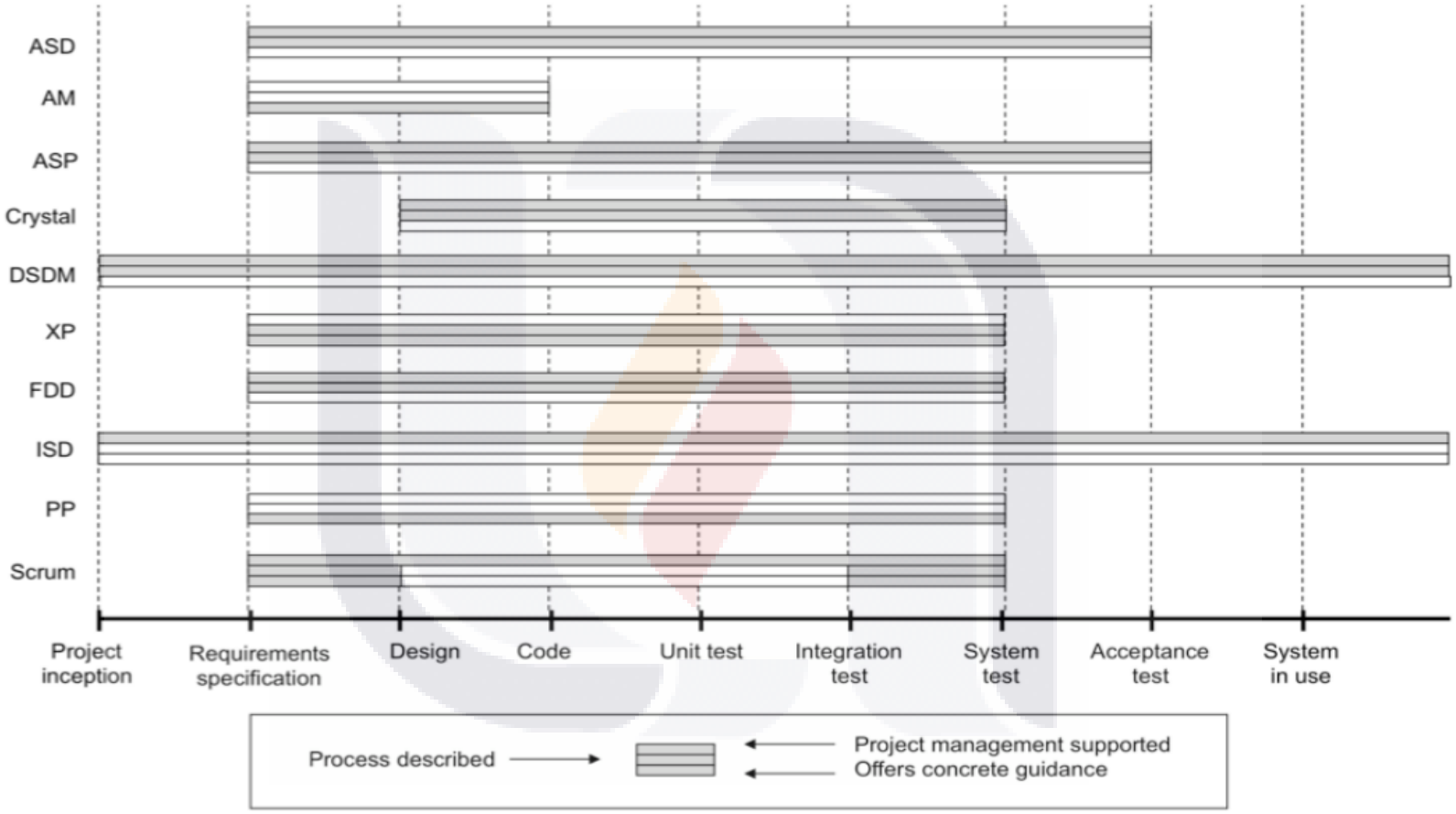


Figure 6 - Comparing project management on agile SDLC methods

Several **Agile Methodologies** acknowledge the high-quality software and customer satisfaction (Javanmard & Alian, 2015), Adaptive Software Development (ASD), Agile Modeling, Crystal Methods, Dynamic System Development, Lean Development, and Scrum are some examples of agile methodologies. Most of the times the agile methodologies must be using combining other practices or methods to cover the whole cycle, Abrahamsson et al (2010) analyzed most of the agile **SDLC** (see Figure 6) trying to find if they can support project management support, a process described and offers concrete guidance. Abrahamsson et al (2010) concluded that the lack of project management and concrete guidance could be a problem for different situations in development phases.

Laanti et al. (2013) recollected some definitions of agile with different authors and found some words that make emphasis from them. Table 7 represents every agile feature found by Laanti and connects to the twelve principles of agile software from the manifesto (Beck et al., 2001).

Table 7 - Agile features related to the 12 agile principles.

Concept	Twelve Principles of Relation	Reference
Effective	Working software is the primary measure of progress. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Cockburn 2001
Steerable	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Cockburn 2001
Rule-based	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Cockburn 2001
Speed	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. Working software is the primary measure of progress. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Anderson 2003, Larman 2003, Schuh 2004, Ambler 2007,

People	Business people and developers must work together daily throughout the project. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Cockburn 2001, Schuh 2004,
Empowerment	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Schuh 2004,
Change	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Schuh 2004
Value	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. Working software is the primary measure of progress.	Schuh 2004
Delivery	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Lyytinen 2006
Innovations	Continuous attention to technical excellence and good design enhances agility.	Lyytinen 2006
Feedback	Continuous attention to technical excellence and good design enhances agility. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Schuh 2004, Subramaniam 2005
Adaptability	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. Continuous attention to technical excellence and good design enhances agility.	Subramaniam 2005
Collaboration	Business people and developers must work together daily throughout the project. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Subramaniam 2005
Iterative	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Ambler 2007, IEEE 2007, Wikipedia 2007
Incremental	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Ambler 2007,

Selforganizing	The best architectures, requirements, and designs emerge from self-organizing teams.	Ambler 2007,
Less processdriven	Working software is the primary measure of progress.	Ambler 2007,
Collaborative	Business people and developers must work together daily throughout the project. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Ambler 2007,
Cost-conscious	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Ambler 2007,
Customer-driven	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Ambler 2007,
Responsiveness	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Larman 2003, Lyytinen 2006, Nerur and Balijepally 2007
Flexibility	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Larman 2003, Nerur and Balijepally 2007
Responsive	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	IEEE 2007
Conceptual framework		Wikipedia 2007

Boehm and Turner (2003) analyzed some characteristics of agile and traditional methods (see Table 8) using the Application, Management, Technical, and Personnel constructors. The differences are clear: Delivering value to the customer as quickly as possible in short increments is the key element of agile methods.

Table 8 - General characteristics for agile methods and traditional methods

Project characteristics	Agile home ground	Plan-driven home ground
Application		
Primary goals	Rapid value, responding to change	Predictability, stability, high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent, high change, project focused	Stable, low change, project and organization focused
Management		
Customer relations	Dedicated onsite customers, focused on prioritized increments	As-needed customer interactions, focused on contract provisions
Planning and control	Internalized plans, qualitative control	Documented plans, quantitative control
Communications	Tacit interpersonal knowledge	Explicit documented knowledge
Technical		

Project characteristics	Agile home ground	Plan-driven home ground
Requirements	Prioritized informal stories and test cases, undergoing unforeseeable change	Formalized project, capability, interface, quality, foreseeable evolution requirements
Development	Simple design, short increments, refactoring assumed inexpensive	Extensive design, longer increments, refactoring assumed expensive
Test	Executable test cases define requirements, testing	Documented test plans and procedures
Personnel		
Customers	Dedicated, colocated Crack (Collaborative, representative, authorized, committed, and knowledgeable) performers	Crack* performers, not always colocated
Developers	At least 30% full-time Cockburn Level 2 and 3 experts; no Level 1B or Level –1 personnel (See the “Cockburn’s Three Levels of Software Understanding)	50% Cockburn Level 3s early; 10% throughout; 30% Level 1B’s workable; no Level –1s (See the “Cockburn’s Three Levels of Software Understanding)
Culture	Comfort and empowerment via many degrees of freedom (thriving on chaos)	Comfort and empowerment via framework of policies and procedures (thriving on order)

Boehm and Turner (2003) also created the five critical agility and plan-driven factors (see Table 9) where is possible to know if projects fit into an agile or a traditional methodology. Finally, Figure 7 shows a polar chart where the five factors can be graphed and provides information about what methodology should be chosen.

Table 9 - The five critical agility and plan-driven factors.

Factor	Agility discriminators	Plan-driven discriminators
Size	Well matched to small products and teams; reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams; hard to tailor down to small projects.
Criticality	Untested on safety-critical products; potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products; hard to tailor down efficiently to low-criticality products.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments but present a source of potentially expensive rework for highly stable environments.	Detailed plans and “big design up front” excellent for highly stable environment, but a source of expensive rework for highly dynamic environments.
Personnel	Require continuous presence of a critical mass of scarce Cockburn Level 2 or 3 experts; risky to use nonagile Level 1B people.	Need a critical mass of scarce Cockburn Level 2 and 3 experts during project definition, but can work with fewer later in the project—unless the environment is highly dynamic. Can usually accommodate some Level 1B people.
Culture	Thrive in a culture where people feel comfortable and empowered by having many degrees of freedom; thrive on chaos.	Thrive in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures; thrive on order.

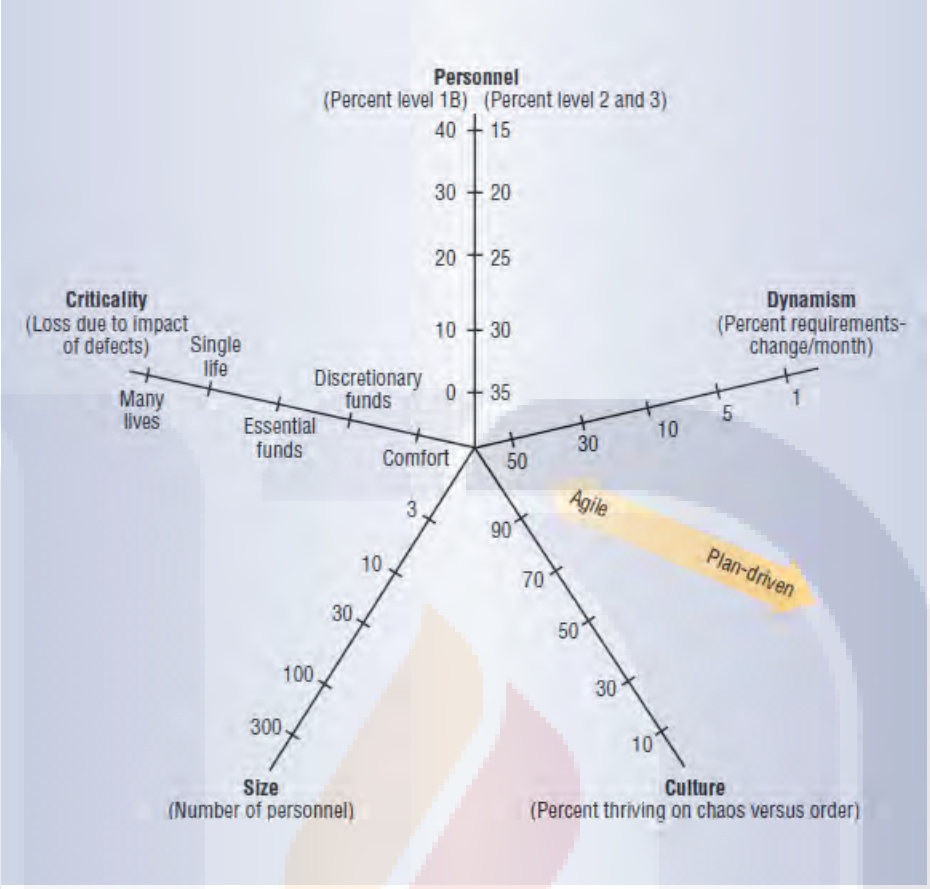


Figure 7 - Polar chart with the five critical factors.

To have success, it is very important to classify the level of expertise for every developer of the team. Cockburn (2002) identified three levels of people that can be sort inside a software method, Boehm and Turner (2003) modified their work splitting level 1 to make difference between Agile and plan-driven methods and added level (see Table 10). Level -1 people should be identified as soon as possible to be reassigned to other activities rather than development. Level 1B people are average and below, with a stable project they can work without any problem, but sometimes can slow the team on urgency changes. Level 1A people can work very well if they are enough people level 2 to guide them. Level 2 people can manage small teams with the guide of Level 3 people, with some experience they can become Level 3. Level 3 people are the most experienced people, able to manage large projects.

Table 10 - People level for software development.

Level	Characteristics
3	Is able to revise a method (break its rules) to fit an unprecedented new situation.
2	Is able to tailor a method to fit a precedented new situation.
1A	With training, is able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, and complex COTS integration). Can become Level 2 with experience.
1B	With training, is able to perform procedural method steps (e.g., coding a simple method, simple refactoring, following coding standards and capability model procedures, and running tests). Can master some Level 1A skills with experience.
-1	May have technical skills, but is unable or unwilling to collaborate or follow shared methods.

Having all above information from Boehm and Turner (2003) on this PhD Dissertation we will try to keep Personnel with 30% 1B and 20% with level 2 or 3. Criticality should be at the middle of the bar or below with projects that have not high impact. Dynamism allows changes up to 5% per month so that the project can keep a balancing. Culture stated at the middle of the bar because an order should be implemented even though we are talking about Agile development. Size of the team should be small and there should be no more than 15 people, if the project needs more people, it must be divided in smaller teams to be more manageable. Figure 8 shows the graphic made by Boehm and Turner (2003) with the approach for our methodology presented on this document.

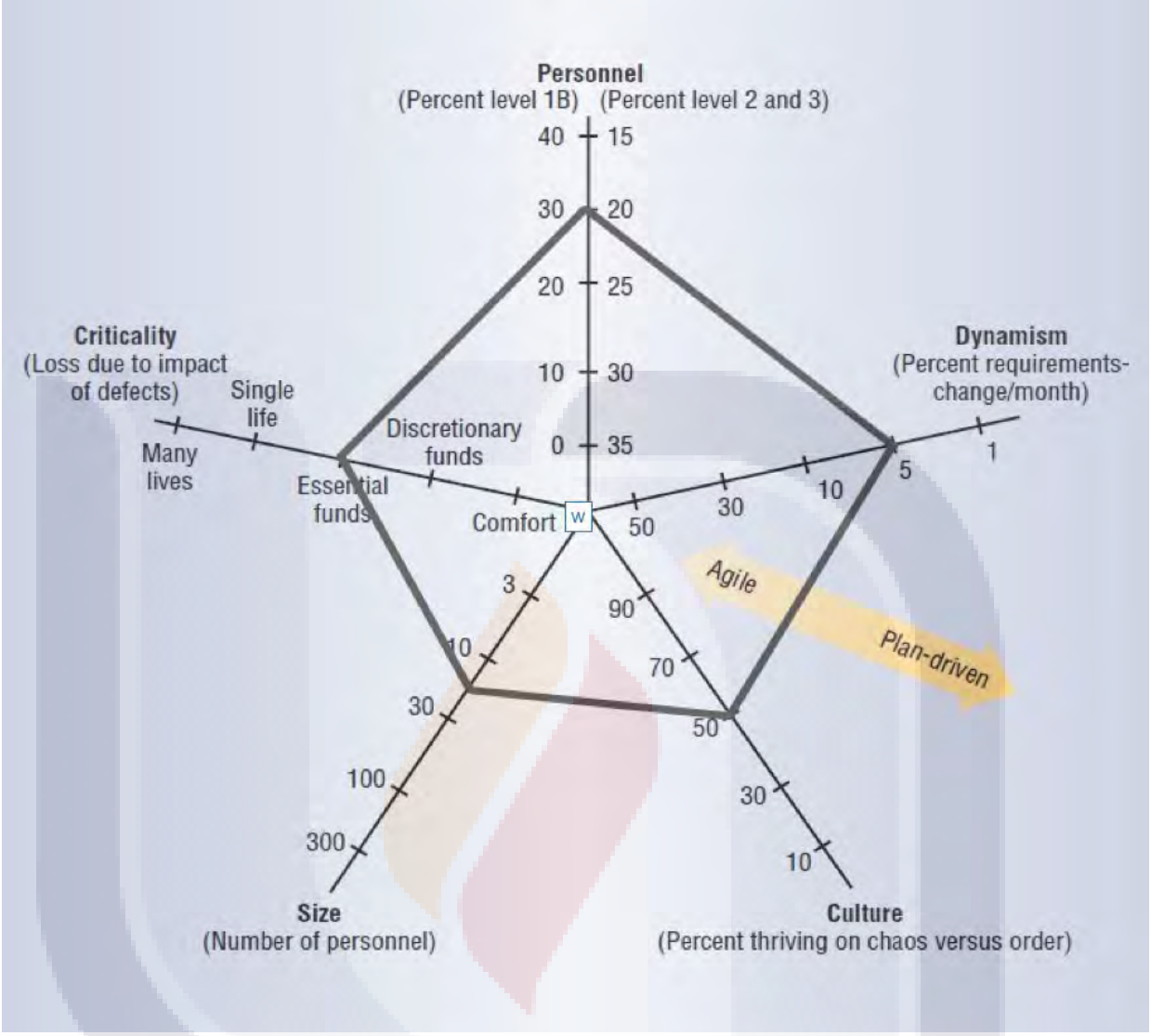


Figure 8 - Boehm and Turner graphic with new methodology approach.

Agile and traditional have unique features, each of them can have one value from ten to one to see the weight that has every feature inside the SDLC. Figure 9 represents a word cloud with the ten most valuable features in Agile development while Figure 10 represents a word cloud with the ten most valuable features inside traditional development.

Figure 9 - Most important features on Agile SDLC.

Figure 10 - Most important features on linear SDLC.

3.1.2.2 REVIEW OF OFFICIAL SCRUM – MAIN AND MOST USE AGILE SDLC

During the State of Agile survey (2021) the 75% of the respondents said that they are using Scrum as Agile framework. Scrum is one of the most popular Agile frameworks because it is very easy to learn and simple. Schwaber and Sutherland (2020) created the Scrum Guide that includes only thirteen pages and explains all the roles, events, and artifacts. Figure 11 shows the Scrum process (Scrum.org, 2021) with all its elements. The Scrum Guide (Schwaber & Sutherland, 2020) defines Scrum as "***a lightweight framework that helps people, teams, and organizations generate value through adaptive solutions for complex problems***".

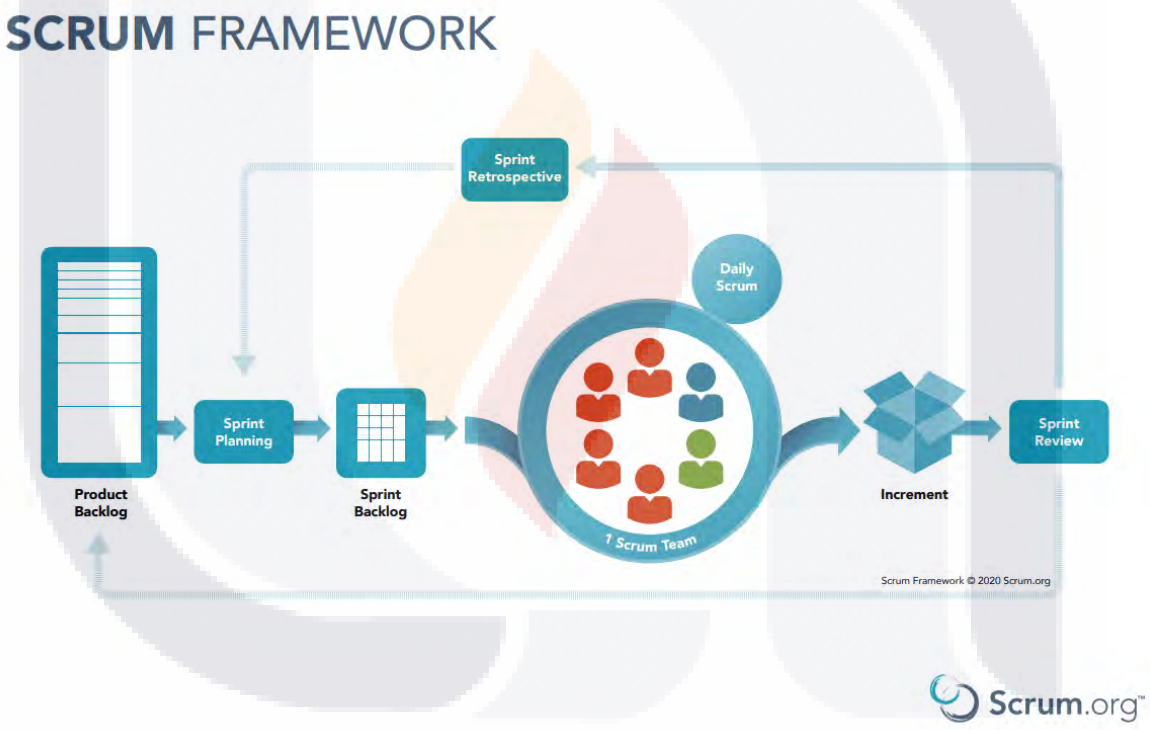


Figure 11 - Scrum Process Diagram with all its elements taken from Scrum.org.

Scrum defines only three roles that work together all the time to provide the max value to the project. The Product Owner is the person who is accountable to maximize the value of every task, this role is in charge of the product backlog where he or she put the tasks to be done sorted by the most important to the less important task. The Developers are the people that have to complete every task put inside the Sprint Backlog, they manage all those tasks and do the plan every Sprint so that they make sure that every iteration has the reasonable tasks to be

developed in time. The Scrum Master is the person who is responsible to implement every Scrum element as it is defined in the Scrum Guide, the Scrum Master also coaches the team and removes every blocker that the developers must continue.

The Scrum artifacts represent the work or the value and are available for everyone to make all the team's work transparent. Product Backlog has all the tasks manage by the Product Owner and are sorted by the most to the less important. The Sprint Backlog is created every Sprint by the developers, they take the number of tasks that can be complete in a certain period, when the Sprint ends, the completed tasks are added to the Increment that is the sum of all completed tasks in previous Sprints, this Increment should be functional.

Finally, every Scrum event "***is a formal opportunity to inspect and adapt Scrum artifacts. These events are specifically designed to enable the transparency required. Failure to operate any events as prescribed results in lost opportunities to inspect and adapt. Events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. Optimally, all events are held at the same time and place to reduce complexity***" (Schwaber & Sutherland, 2020). The Sprint is where all the work is done, it has a fixed timeframe, every time that a Sprint is over a new one Starts immediately. Sprint Planning starts every Sprint and it's a meeting where the developers plan the work to be done during the timeframe choosing the most important tasks from the Product Backlog and create a Sprint Backlog. Daily Scrum is a fifteen-minute meeting that occurs every day to "***improve communications, identify impediments, promote quick decision-making, and consequently eliminate the need for other meetings***" (Schwaber & Sutherland, 2020). Sprint Review occurs when the Sprint is almost over, the Scrum Team shows the results of their work and the progress to the Stakeholders. Sprint Retrospective is where the Scrum Team "***identifies the most helpful changes to improve its effectiveness***" (Schwaber & Sutherland, 2020), the feedback from the team is very valuable to enhance the quality of the "***individuals, interactions, processes, tools, and their Definition of Done***" (Schwaber & Sutherland, 2020).

Table 11 shows every Scrum element described above divided into Roles, Events, and Artifacts.

Table 11 - Scrum elements.

Roles	Events	Artifacts
Product Owner	Sprint	Product Backlog
Scrum Master	Sprint Planning	Sprint Backlog
Scrum Team	Daily Scrum Meeting	Product Increment
	Sprint Review	
	Sprint Retrospective	

3.1.2.3 REVIEW OF A ROBUST SCRUM

The official Scrum Guide (Schwaber & Sutherland, 2020) is a good start to learn the Scrum principles, roles, events, and artifacts but on the other hand, it is difficult to start a new project just with the knowledge provided by that guide.

SCRUMstudy organization created an SBOK Guide (2013) as a "***guide for organizations and project management practitioners who want to implement Scrum, as well as those already doing so who want to make needed improvements to their processes. It is based on experience drawn from thousands of projects across a variety of organizations and industries. The contributions of many Scrum experts and project management practitioners have been considered in its development."***

The SBOK Guide (2013) is composed of six phases: Initiate, Plan and Estimate, Implement, Review and Retrospect, and Release. Every single phase has its processes. Table 12 shows the phases and the processes proposed by the SBOK Guide.

Every single element from Scrum is included in the SBOK Guide (2013) and every process is explained step by step so that it is easier to start working with Scrum in a more organized way.

Schwaber (1997) proposed three different phases for Scrum called Pregame, Game, and Postgame. Every single had the different Scrum processes defined in 1997 that are very similar to the current Scrum processes. Figure 12 displays the Scrum Methodology defined by Schwaber (1997).

Table 12 - SBOK Guide phases and processes.

Phase	Processes
Initiate	Create Project Vision Identify Scrum Master and Stakeholder(s) Form Scrum Team Develop Epic(s) Create Prioritized Product Backlog Conduct Release Planning
Plan and Estimate	Create User Stories Approve, Estimate, and Commit User Stories Create Tasks Estimate Tasks Create Sprint Backlog
Implement	Create Deliverables Conduct Daily Standup Groom Prioritized Product Backlog
Review and Retrospect	Convene Scrum of Scrums Demonstrate and Validate Sprint Retrospect Sprint
Release	Ship Deliverables Retrospect Project

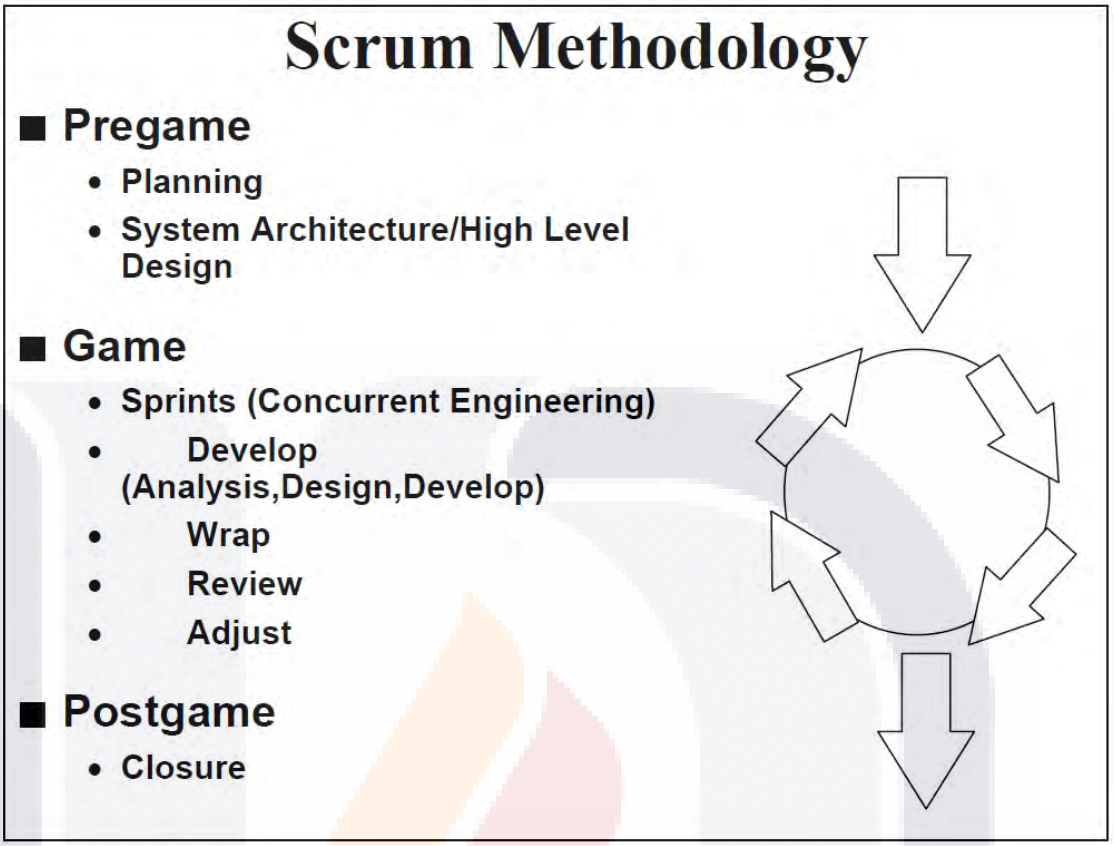


Figure 12 - Scrum Methodology created by Schwaber in 1997.

Using the SBOK Guide (2013) and following the Schwaber (1997) proposal, we are proposing three different phases: Pregame, Game, and Postgame. Every single phase is composed of different processes that help to maintain order with the agility that we need.

We can match the three phases with the phases inside eXtreme Programing Agile Methodology proposed by Dudziak (1999): Exploration, Releases Planning, Iteration Planning, Implementation, Functional Test, and Release. Figure 13 shows the eXtreme Programing simplified structure created by Dudziak (1999).

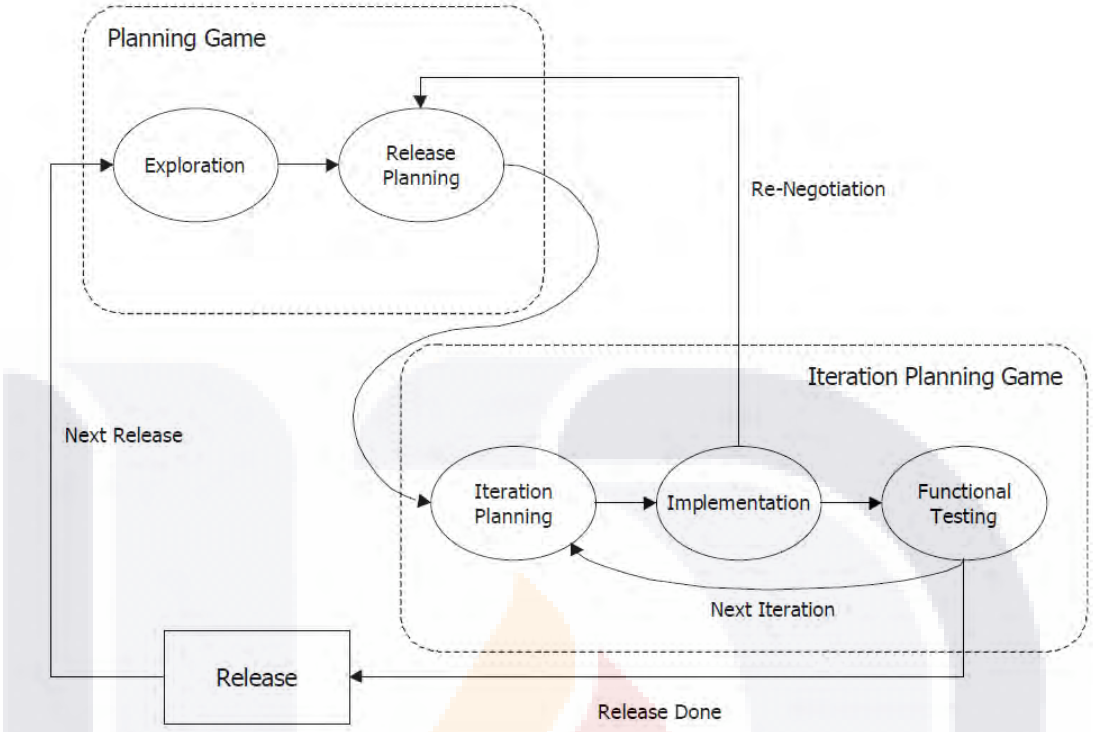


Figure 13 - eXtreme Programing simplified process structure.

To match Scrum and eXtreme Programing phases is it possible to rename Planning Game as Pregame, Iteration Planning Game as Game, and Release as Postgame. Table 13 displays the three proposed phases with their processes and the corresponding eXtreme Programing phase. It is also possible to use the Implementation phase where the user stories are developed.

The Pregame phase is where basics are created, the Create Project Vision process is conducted by the Product Owner so that it will provide inspiration and focus for the whole project. Develop Epics is a process where all the team meets to create appropriate Epics for the project. Once the Epics are created it is necessary to follow the Created Prioritized Product Backlog where the Epics are refined, elaborated, and sorted from the most valuable to the less valuable. Conduct Release Planning process is where the length of the Sprint is defined, the Release Planning Schedule is created, and the deployment scheduled can be shared with stakeholders.

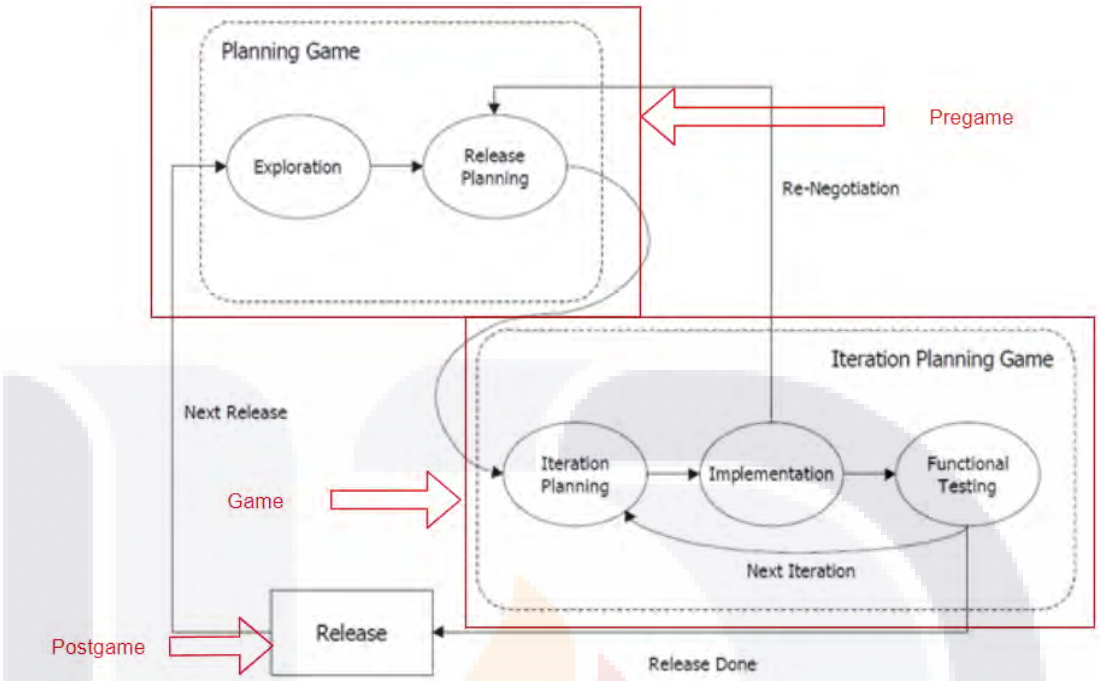


Figure 14 - eXtreme Programing Phases renamed.

The Game phase is the iterative part of the project where every Sprint starts and ends with all Scrum ceremonies. Create Sprint Backlog process runs when the new Sprint starts and provides the necessary tasks to complete during that period. Conduct Daily Standup process is the daily Scrum ceremony where every Scrum Team member updates their process and report any impediment. The Groom Prioritized Product Backlog process is vital for having a healthy Sprint Backlog for the next Sprints, is where the team meets to update and maintain the Product Backlog. Build Sprint Increment process is the time where developers work on user stories and the development is done. Demonstrate and Validate Sprint process is where the Sprint Review ceremony is conducted to show the progress from the team during that Sprint. Finally, the Retrospective Sprint process has the Sprint Retrospective ceremony so that the team can identify opportunities areas to improve during the next Sprints.

The Postgame phase represents the deliverables where the increment is delivered and deployed. This phase only has the Ship Deliverables process. Figure 14 represents this approach.

Figure 15 displays a more robust Scrum process using the three defined phases and their processes. Please note that the Game phase is the iterative part that must be repeated until the project is done.

Table 13 - Phases, processes, roles, and artifacts used for a new methodology.

			Roles		
Phase	**eXtreme Programing phase**	**Processes**	**Principal**	**Support**	**Artifacts**
Pregame	Exploration	Create Project Vision	Product Owner	Scrum Master	Project Vision Statement
		Develop Epics	Product Owner	Scrum Master Scrum Team	Product Backlog
		Create User Stories	Product Owner	Scrum Master Scrum Team	User Stories
	Release Planning	Created Prioritized Product Backlog	Product Owner	Scrum Master Scrum Team	Prioritized Product Backlog
		Conduct Release Planning	Product Owner	Scrum Master Scrum Team	Release Planning Schedule
Game	Iteration Planning + Implementation + Functional Testing	Create Sprint Backlog	Scrum Team	Product Owner Scrum Master	Sprint Backlog
		Conduct Daily Standup	Scrum Team	Product Owner Scrum Master	Kanban board
		Build Sprint Increment	Scrum Team	Scrum Master	Updated User Stories
		Demonstrate and Validate Sprint	Scrum Team	Product Owner Scrum Master	Increment
		Retrospective Sprint	Scrum Team	Product Owner Scrum Master	Agreed Actionable Improvements
Postgame	Release	Ship Deliverables	Scrum Team	Scrum Master	Working Deliverables Agreement

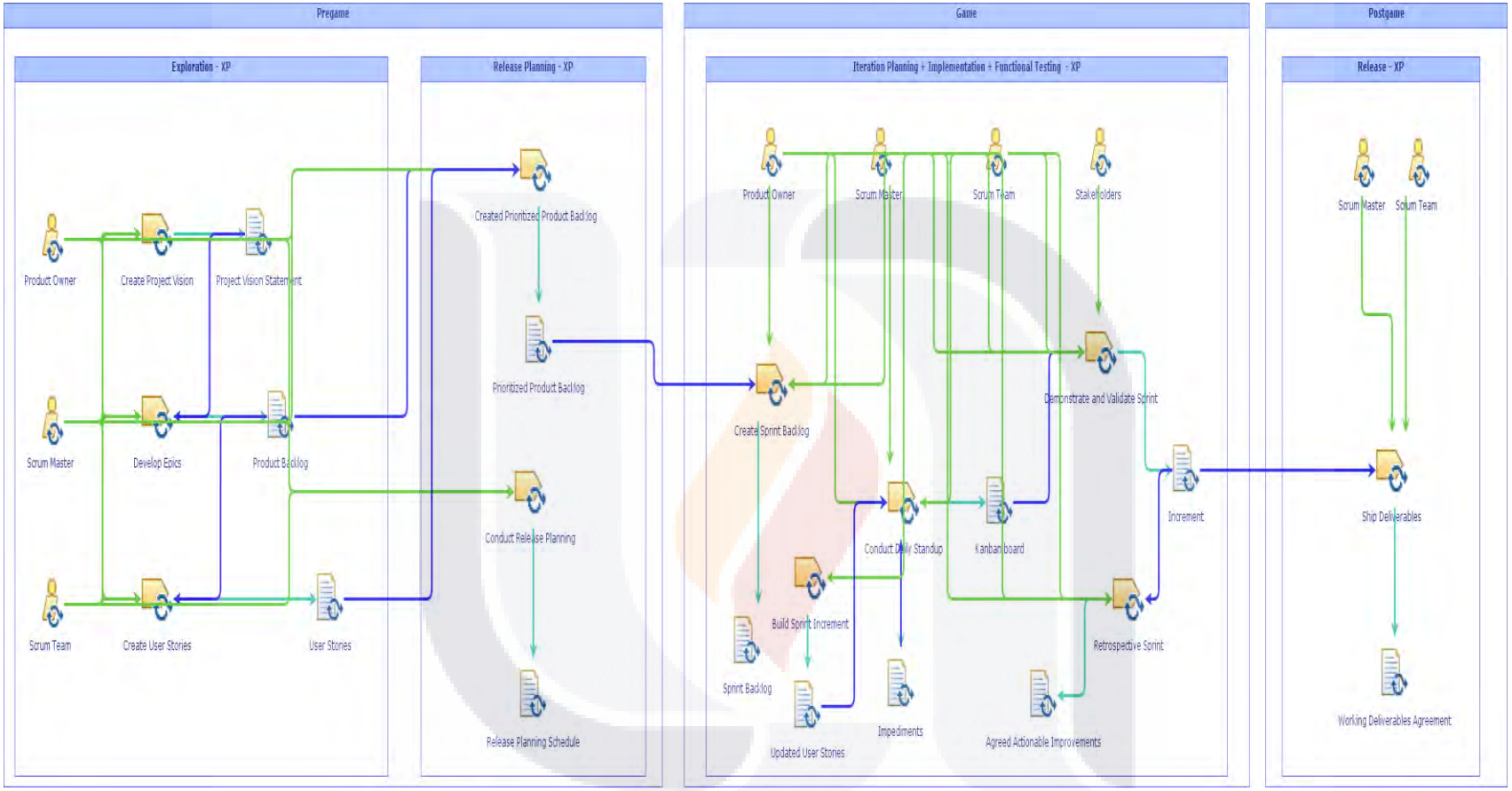


Figure 15 - More robust Scrum process.

Hence, this section provides the following important concepts:

Scrum

"***Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems***." (Schwaber & Sutherland, 2020, p. 3)

Scrum Team

"***The fundamental unit of Scrum is a small team of people, a Scrum Team. The Scrum Team consists of one Scrum Master, one Product Owner, and Developers. Within a Scrum Team, there are no sub-teams or hierarchies. It is a cohesive unit of professionals focused on one objective at a time, the Product Goal***." (Schwaber & Sutherland, 2020, p. 5)

Product Owner

"***The Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. How this is done may vary widely across organizations, Scrum Teams, and individuals***." (Schwaber & Sutherland, 2020, p. 5)

Scrum Master

"***The Scrum Master is accountable for establishing Scrum as defined in the Scrum Guide. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization***." (Schwaber & Sutherland, 2020, p. 6)

Developers

"***Developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint***." (Schwaber & Sutherland, 2020, p. 5)

Sprint

"***Sprints are the heartbeat of Scrum, where ideas are turned into value. They are fixed length events of one month or less to create consistency. A new Sprint starts immediately after the conclusion of the previous Sprint***." (Schwaber & Sutherland, 2020, p. 7)

Sprint Planning

"***Sprint Planning initiates the Sprint by laying out the work to be performed for the Sprint. This resulting plan is created by the collaborative work of the entire Scrum Team***." (Schwaber & Sutherland, 2020, p. 8)

Daily Scrum

"***The purpose of the Daily Scrum is to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work***." (Schwaber & Sutherland, 2020, p. 9)

Sprint Review

"***The purpose of the Sprint Review is to inspect the outcome of the Sprint and determine future adaptations. The Scrum Team presents the results of their work to key stakeholders and progress toward the Product Goal is discussed***." (Schwaber & Sutherland, 2020, p. 9)

Sprint Retrospective

"***The purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness***." (Schwaber & Sutherland, 2020, p. 10)

Product Backlog

"***The Product Backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team***." (Schwaber & Sutherland, 2020, p. 10)

Sprint Backlog

"***The Sprint Backlog is composed of the Sprint Goal (why), the set of Product Backlog items selected for the Sprint (what), as well as an actionable plan for delivering the Increment (how).***" (Schwaber & Sutherland, 2020, p. 11)

Increment

"***An Increment is a concrete stepping stone toward the Product Goal. Each Increment is additive to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be usable.***" (Schwaber & Sutherland, 2020, p. 11)

eXtreme Programming (XP)

"**XP is also a lightweight methodology or what Alistair Cockburn calls a "Crystal Methodology". In short, methodologies of this family have high productivity and high tolerance. Communication is usually strong with short paths, especially informal (not documented). There the is only a small range of deliverables (artifacts), but these are delivered frequently (releases). Processes of the Crystal family identify only a few roles and activities.**" (Dudziak, 1999, p. 4)

3.1.3 ON BUSINESS PROCESS MANAGEMENT SYSTEMS (BPMS) DEVELOPMENT PLATFORMS, METHODOLOGIES, AND SOFTWARE APPLICATIONS

3.1.3.1 CORE DEFINITIONS (BPMS, WORKFLOW MANAGEMENT SYSTEMS, BPMS DEVELOPMENT PLATFORMS, BPMS SOFTWARE APPLICATION/PAIS)

Before going deeper into BPMS and its methodologies, it is important to have a clear definition of the most important elements that will be used in this Ph.D. Dissertation from now on.

In the decade of the nineties, there were Workflow Management Systems (WFMS) that helped to integrate existing applications and isolate the management of the business process into another component. The Workflow Management Coalition (WfMC), cited by van der Aalst *et al*. (2003), defined WFMS as "***A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications***". van der Aalst *et al*. (2003) refer to the WFMS as the essence of BPMS even Reijers (2006) uses WFMS and BPMS as synonymous.

Karagiannis (1995, p. 10) defines the BPMS as "***Information systems dealing with the definition, administration, customization, and evaluation of tasks evolving from business processes as well as from organizational structures are called Business Process Management Systems.***" Business Process Management Initiative (BPMI), cited by Jung *et al*. (2007, p. 22), says that BPMS "***is to integrate systems, automate routine activities, manage all phases of processes, deploy process seamlessly, and provide end-to-end visibility and control***". Finally, Reijers (2006, p. 390) described a BPMS as a "***piece of generic software that supports activities such as the modeling, analysis and enactment of business processes."***

Mutschler *et al*. (2008) divide the information systems into Process-oriented Information Systems (POIS) and Process-aware Information Systems (PAIS). POIS are developed taking into account the process of the company so that those information systems (IS) could be obsoleted once the processes are updated. On the other hand, PAIS "***does not contain any information about the structure and the processes of a particular organization. Instead, an organization needs to configure the PAIS by specifying processes, organizational entities,***

and business objects" (Mutschler et al., 2008, p. 7). The PAIS was defined as "***a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of processes models***" by Dumas *et al*. (2005, p. 7).

Krafzig *et al*. (2005) refer that BPMS projects can be developed with a standard software development methodology while Ravesteyn and Batenburg (2010a, p. 2) signs that "***standard software development methodologies, however – such as the waterfall method, rapid application development or rational unified process – ignore the business or organizational aspects."***

In this Ph.D. dissertation, we are taking BPMS/PAIS as a "***Software system for supporting the operation and monitoring of a full Business Process."*** (adapted from Reijers, 2006, p. 390)***.*** We also consider WFMS as the heart of BPMS, in this modern era both are synonymous. BPMS can be also a Process-aware Information system (PAIS) because they do not have any hardcoded logic for the organization and the customer is the one that must configure every aspect of his organization.

With BPMS already defined, it is important to know that a platform is defined as "***a bundle of functions that can serve as the basis of certain applications whose value changes over time***" (Taudes et al., 2000, p. 228) so that a BPMS/PAIS Development Platform can be defined as **"Software development platform used for designing, building, running, and monitoring a BPMS/PAIS."**. Finally, a BPMS Application is the information system developed with the help of a BMPS Development Platform.

With all the main terms defined we can divide BPMS into three main important elements: **BPMS (PAIS) Business Methodologies** that can be classic or agile, **BPMS (PAIS) Business Development Platforms** that can be modern or classic, and **BPMS (PAIS) Business Software Applications** that can be developed for small enterprises, medium enterprises, and large enterprises. Figure 16 displays a conceptual map that represents the BPMS (PAIS) division. The main objective of this work is to help small enterprises with low-budget.

It is important to know the difference between a Business Process Platform and a Business Software Application; The first is a piece of generic software where the developer creates all the business process workflow and configures the data, interfaces, and functions to create a Business Software Application. This application will be used by the final users and will have all the functionality, data, and processes already configured by the developers.

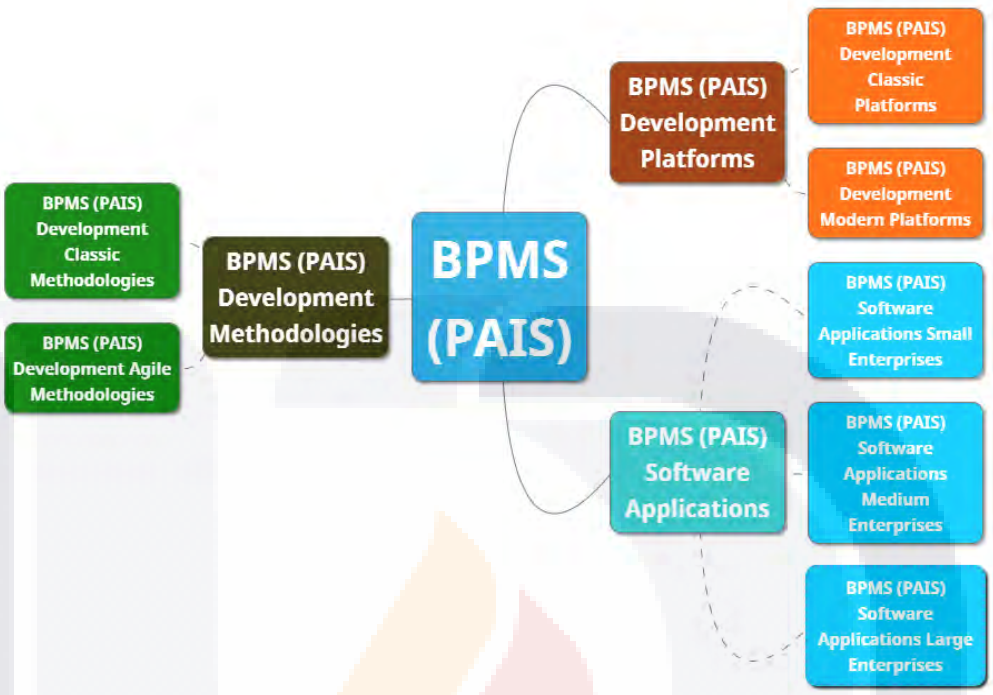


Figure 16 - BPMS (PAIS) division in three main categories.

3.1.3.2 REVIEW OF BPMS PLATFORMS – OPEN SOURCE VS COMMERCIAL

Being able to be defined BPMS now it is important to clearly define Low-code platforms. The term “low-code” was referred to by Richardson and Rymer (2014, p. 1) as “***application platforms that accelerate app delivery by dramatically reducing the amount of hand-coding required. Faster delivery is the primary benefit of these application platforms; they also help firms respond more quickly to customer feedback after initial software releases and provision mobile and multichannel apps. Usage of low-code platforms is gaining momentum for customer-facing applications***”. While Waszkowski (2019, p. 1) stated that Low-Code “***Programming enables the programmer to spend less time thinking about the syntax of the code and to put more emphasis on designing the aesthetics and functionality of the application, so reducing the amount of time spent on troubleshooting and implementing***”. Nowadays Low-code development platforms can be local or cloud-based so that it is possible for the development, and deployment of functional software with minimal or no code (Sahay et al., 2020).

As a cloud-based environment, the low-code platforms have an architecture where everything is connected to a server where it handles the calls to different internal and external services, repositories, databases, compilers, code generators, and optimizers. Sahay *et al*. (2020) represented the typical low-code platform architecture as shown in Figures 17 and Figure 18 for traditional and cloud-based low-code platforms.

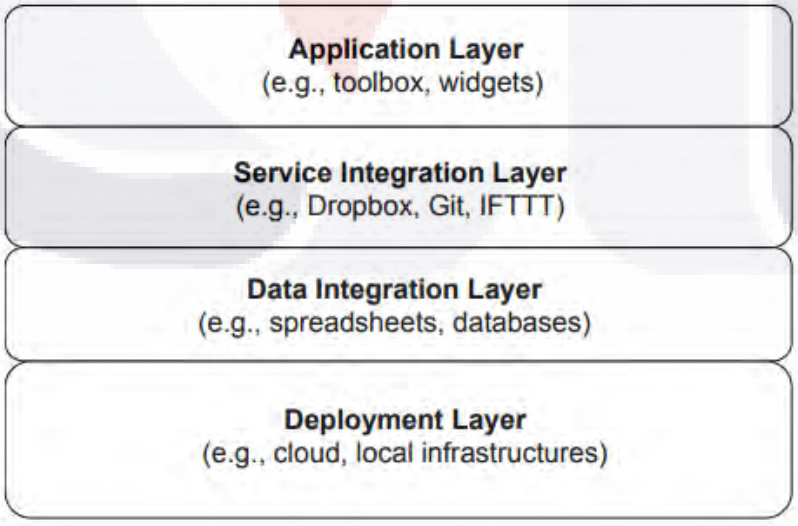


Figure 17 - Low-code platform typical architecture.

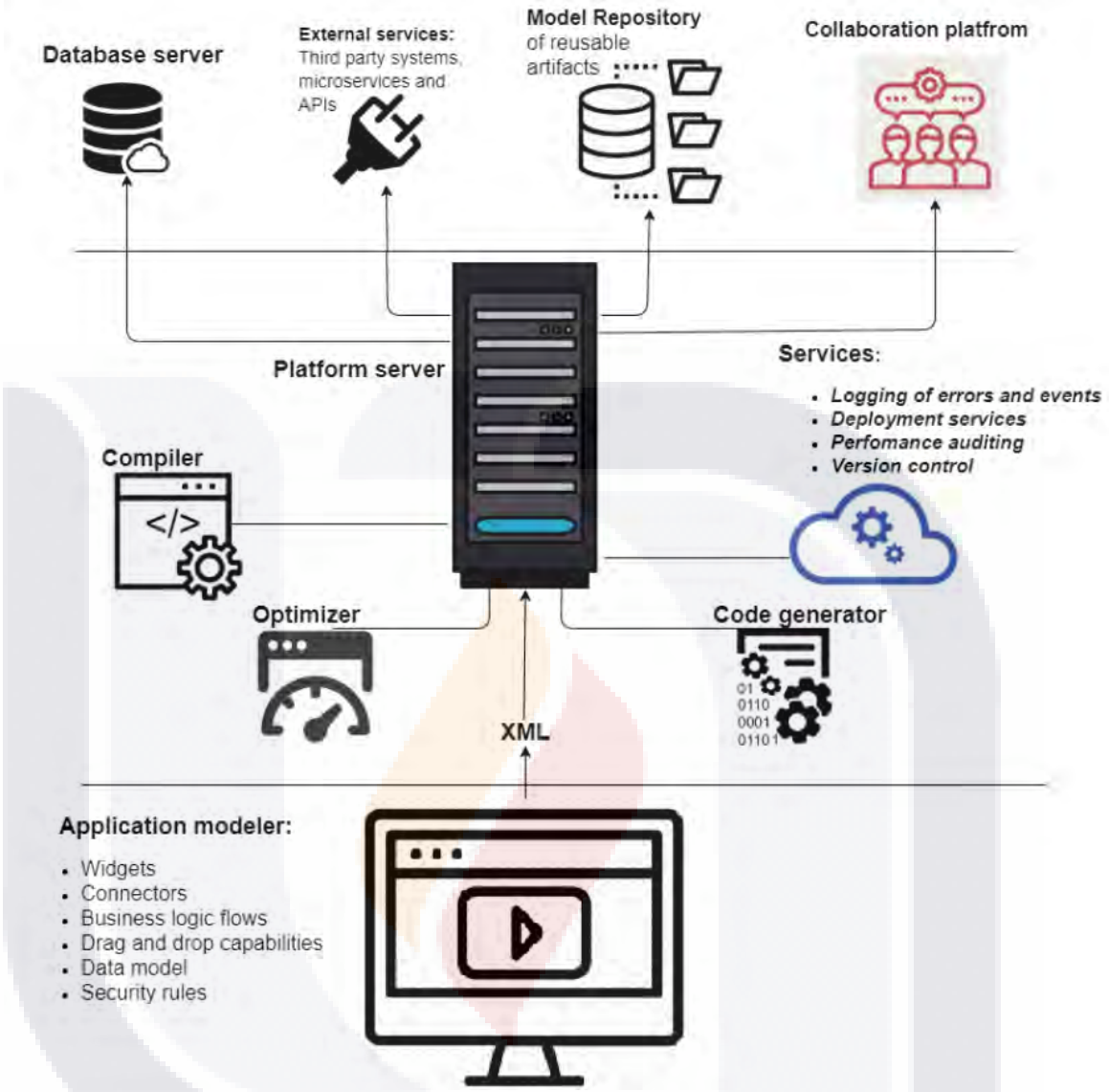


Figure 18 - Low-code platform cloud-based architecture.

There are commercial and open-source Low-code platforms so we are going to compare three of the most important platforms in both categories. Using Magic Quadrant for Enterprise Low-Code Application Platforms by Gartner (2019) we will take three low-code platforms from the LEADERS quadrant that are Appian, OutSystems, and Mendix.

Sahay *et al*. (2020) created a taxonomy for Low-Code Development Platforms that compared different platforms with different constructors like graphical user interface, interoperability support, security support, collaborative development support, reusability support, scalability, business logic specification mechanisms, application build mechanisms, deployment support, and kinds of supported

applications. Table 14 displays the work done by Sahay *et al*. (2020) comparing Appian, OutSystems, and Mendix.

For Low-Code Platforms open-source it is needed to take another approach, we cannot compare with the same parameters used in Table 3-11 because used to have fewer functionalities with the great advantage that are free and can be used by small enterprises with low-budget that are our main objective for this work. To compare open-source platforms we use Mora *et al.* (2016) work that compares open-source elements based on Risks Categories like Financial, Organizational, End User, and Technical. The work created a tool called Multi-Attribute Decision Making (MADM) after evaluating 12 frameworks: Capgemini Open Source Maturity Model, Navica Open Source Maturity Model (OSMM), Open Business Readiness Rating (OpenBRR), Open Business Quality Rating (OpenBQR), Quality Model for Open Source Selection (QMOSS), QualOSS, Software Quality Observatory for Open Source Software model (SQO-OSS), OpenSource Maturity Model (OMM), QualiPSo—Quality Platform for Open Source Software, IRCA Model, Method for Qualification and Selection of Open Source Software (QSOSv2), and the Evaluation Framework for Free/Open Source Projects (EFFORT). Table 15 represents Mora's comparative table for open-source platforms.

In this document, we will focus on open-source low-code development platforms so that the user cannot be worried about rising prices, having no control of the code, and unexpected platforms shut down even though commercial platforms can offer advanced functionalities (Luo et al., 2021).

Table 14 - Low-code commercial platforms compared by Sahay et al.

Feature	OutSystems	Mendix	Appian
Graphical user interface			
Drag-and-drop designer	Yes	Yes	Yes
Point and click approach	No	No	No
Pre-built forms/reports	Yes	Yes	Yes
Pre-built dashboards	Yes	No	No
Forms	No	No	No
Progress tracking	Yes	Yes	Yes
Advanced reporting	No	No	No
Built-in workflows	No	No	No
Configurable workflows	No	No	No
Interoperability support			
Interoperability with external service	Yes	Yes	Yes
Connection with data sources	Yes	Yes	Yes
Security Support			

Feature	OutSystems	Mendix	Appian
Application security	Yes	Yes	Yes
Platform security	Yes	Yes	Yes
Collaborative development support			
Off-line collaboration	Yes	Yes	Yes
On-line collaboration	Yes	Yes	Yes
Reusability support			
Built-in workflows	No	No	No
Pre-built forms/reports	Yes	Yes	Yes
Pre-built dashboards	Yes	No	No
Scalability			
Scalability on number of users	Yes	Yes	Yes
Scalability on data traffic	Yes	Yes	No
Scalability on data storage	Yes	Yes	No
Business logic specification mechanisms			
Business rules engine	Yes	Yes	Yes
Graphical workflow editor	Yes	Yes	No
AI enabled business logic	Yes	No	Yes
Application build mechanisms			
Code generation	Yes	No	No
Models at run-time	No	Yes	Yes
Deployment support			
Deployment on cloud	Yes	Yes	Yes
Deployment on local infrastructures	Yes	Yes	Yes
Kinds of supported applications			
Event monitoring	Yes	Yes	Yes
Process automation	Yes	No	Yes
Approval process control	No	No	No
Escalation management	No	No	No
Inventory management	Yes	Yes	Yes
Quality management	No	Yes	Yes
Workflow management	Yes	Yes	Yes

Table 15 - Open-source comparative table based on risk.

Risk Attributeure	Definition	Risk Category	Null risk value	Low-risk value	Moderate risk value	Hight risk value	Certain risk value
New business opportunity	Extent of introducing an innovative business process supported by the tool.	Financial	Very high	High	Moderate	Low	Very low
Switching costs	Extent of overall costs caused for the FLOSS adoption.	Financial	Very high	High	Moderate	Low	Very low
Training	Availability of free or affordable user and technical courses.	Organizational	Very high	High	Moderate	Low	Very low
Top management support	Extent of the economic and political support from the highest level of management.	Organizational	Very high	High	Moderate	Low	Very low
Internal expertise	Existence of FLOSS expertise in the organization.	Organizational	Very high	High	Moderate	Low	Very low
Functionality - quality	Extent of expected and enhanced functionalities provided by the tool.	End user	Very high	High	Moderate	Low	Very low
Usefulness - relevance	Extent of advantage is relatively perceived by users of the FLOSS tool.	End user	Very high	High	Moderate	Low	Very low
Usability	Easiness of installation, learning and utilization of the tool.	End user	Very high	High	Moderate	Low	Very low
Community support	Availability of technical support for tool utilization.	Technical	Very high-low cost	High-low cost	Sufficient-high cost	Scarce-high cost	None-high cost
Documentation	Availability of technical and user manuals and extra documents.	Technical	Very high	High	Moderate	Low	Very low
Maturity – longevity	Period of first release of tool.	Technical	Decades	Several years	One year	Few months	One month
Security - reliability	Extent of error-free status and hiddenflaws of the tool.	Technical	Very high	High	Moderate	Low	Very low

The Low-code open-source development platforms chosen for this work are Joget, jBPM, and Camunda. The comparison using the Mora *et al.* (2016) tool Multi-Attribute Decision Making (MADM) based on risks is displayed in Table 15.

Table 16 - Comparison between Low-code open-sour development platforms using the MADM tool.

Risk Attribute	Risk Category	Joget	jBPM	Camunda
New business opportunity	Financial	High	High	High
Switching costs	Financial	Low	Moderate	Very high
Training	Organizational	Low	Moderate	Moderate
Top management support	Organizational	Moderate	Moderate	Moderate
Internal expertise	Organizational	Very high	Very high	Very high
Functionality - quality	End user	Low	Moderate	Very low
Usefulness - relevance	End user	Moderate	Moderate	Moderate
Usability	End user	Moderate	Moderate	Moderate
Community support	Technical	Low	Moderate	High
Documentation	Technical	Low	High	Moderate
Maturity – longevity	Technical	Very low	High	Moderate
Security - reliability	Technical	Low	High	Low

Using an open-source Decision Making Support software named “Open Decision Maker” it is possible to insert the attributes and their values so that we can have the answer to which Low-code Development Platform will work better for our needs using the values 1 to 9 from against every alternative if number 1 is used it means that both alternatives are equal in that property. To keep the consistency of Table 16 there will be added two numbers to represent every difference between the scale of the table. It means that if Joget vs Camunda on Documentation property is Low vs Moderate the Joget part will be on value 3 for representing one value from the table scale of difference. The risks have three different categories Organizational Risks, End-User Risks, and Technical Risks We weighted Technical Risks with a value of 50% while End-User Risks and Organizational Risks are getting a value of 25%. We consider that Community support, Documentation, Maturity – Longevity, and Security - Reliability play a key role in the decision-making but could be variable according to every project's needs. Figure 19 displays the Open Decision Maker tool with the values provided to the FUNCTIONALITY - QUALITY property.

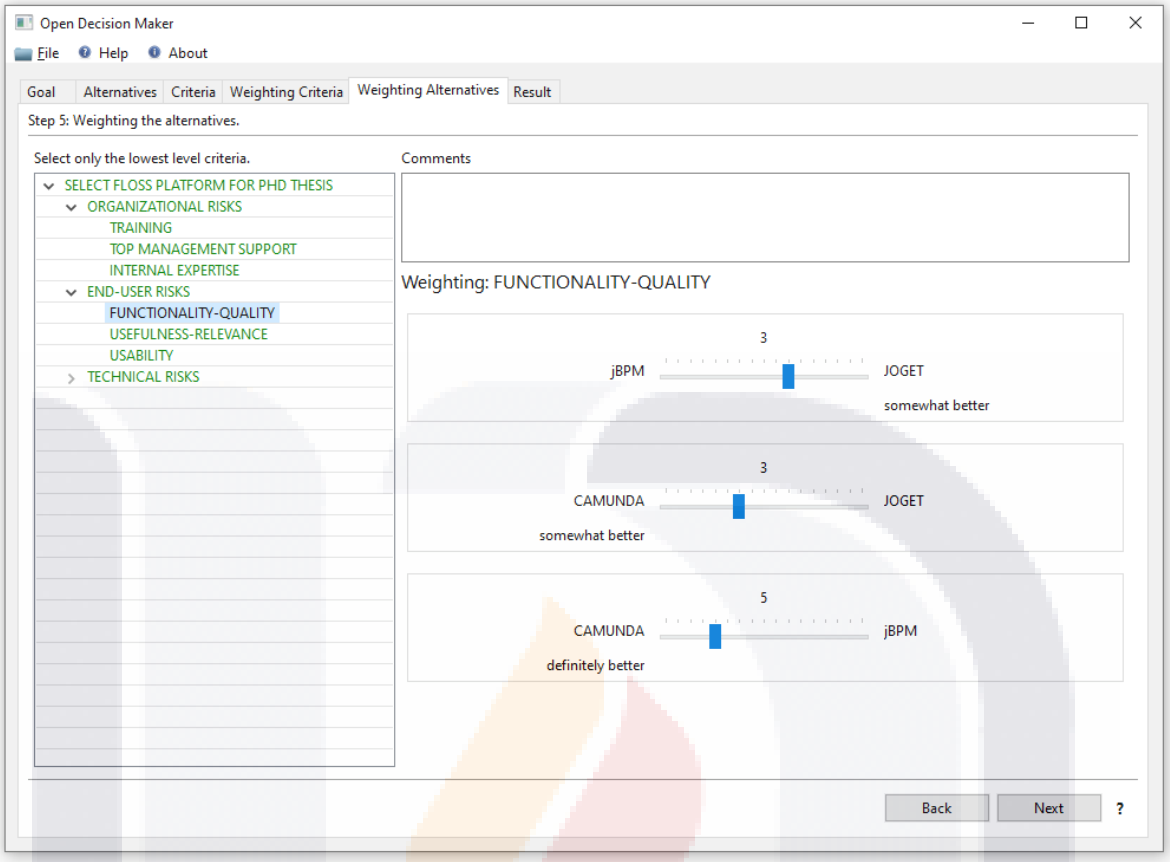


Figure 19 - Open Decision Maker tool with values for the FUNCTIONALITY – QUALITY property.

By getting the results from the tool it is possible to see Joget as the winner with a value of 57.46%, Camunda gets second place with 25.62%, and jBPM with a value of 16.92%. The consistency ratios have a limit value of 0.1 and the values that got a small CR value are FUNCTIONALITY – QUALITY with a value of 0.0332, COMMUNITY SUPPORT with a value of 0.0332, DOCUMENTATION with a value of 0.0332, and MATURITY – LONGEVITY with 0.0559. Figures 20 and 21 display the results tab and the sensitivity analysis from the tool.

Following the results returned by the tool, we can see that using the Low-code open-source Development Platform Joget can reduce the risk due to the low costs, community support, documentation, and maturity of the project. In this Ph.D. dissertation, we are going to use Joget in the next chapters.

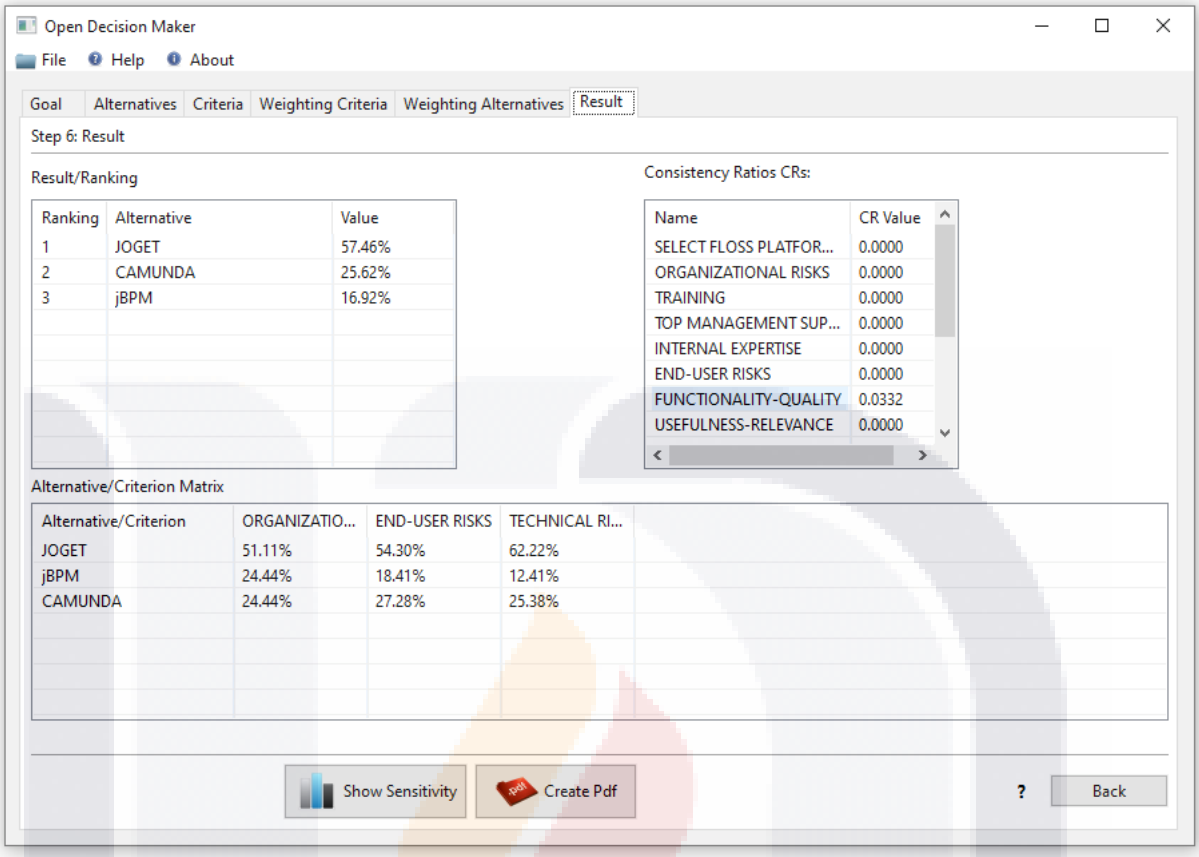


Figure 20 - Result tab that shows Joget as the best Low-code Development Platform.

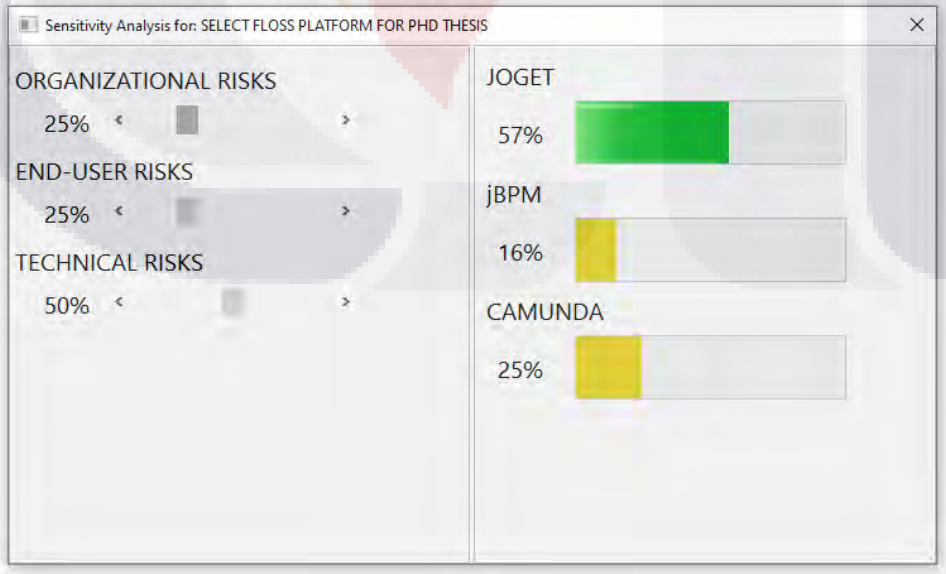


Figure 21 - Sensitivity Analysis with the values of 50% for ORGANIZATIONAL RISKS and 25% for END – USER RISKS and TECHNICAL RISKS.

Hence, this section provides the following important concepts:

Workflow Management Systems (WFMS)

"***A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications***". (van der Aalst *et al*., 2003)

Business Process Management System (BPMS) - Process-Aware Information System (PAIS)

"***Software system for supporting the operation and monitoring of a full Business Process."*** (adapted from Reijers, 2006, p. 390)

BPMS/PAIS Development Platform.

Software development platform used for designing, building, running, and monitoring a BPMS/PAIS.

Low-code

"***Application platforms that accelerate app delivery by dramatically reduce the amount of hand-coding required. Faster delivery is the primary benefit of these application platforms; they also help firms respond more quickly to customer feedback after initial software releases and provision mobile and multichannel apps. Usage of low-code platforms is gaining momentum for customer-facing applications***" (Richardson and Rymer, 2014)

3.1.3.3 BPMS/POIS SYSTEMATIC SELECTIVE LITERATURE REVIEW*

A SSLR differs from Systematic Literature Review (SLR) (Kitchenham et al. 2009) and Mapping Study (MS) (Petersen et al. 2008) research methodologies in the objective of the research, the scope of the research questions, the vastness of the sources of information sought, as well as the studies analyzed (Paré et al. 2015; Boell and Cecez-Kecmanovic 2015). SLR and MS methodologies are usually suitable for specific and mature research topics on which there is extensive literature. SLR pursues quantitative evidence summative purpose on research questions of narrow scope covering exhaustively all available studies – filtered by predefined inclusion and exclusion criteria – and usually are based on many studies. MS is similar to SLR but MS adds a visual multidimensional classification of topics regarding the dimensions of interest for the researchers. In contrast, SSLR method can be used to identify seminal studies on research fields still under maturation or reactivated after a diminished period of research. These seminal studies can be complemented with the selection of illustrative applied studies– also filtered by predefined inclusion and exclusion criteria –, and usually are based on a small number of studies and sources of studies. SLR and MS research methods, thus, are suitable for mature specific research topics where a vast literature on them exists, whereas SSLR is appropriate for research topics still under development or renewed.

Based in Cooper (1988), a Literature Review research methodology can be customized by defining its goal pursuing integration, critique or identification of central of toral issues; its focus on theoretical findings or empirical practices; its perspective as neutral or positional; its coverage through systematic exhaustive search, systematic selective search or representative search; its organization based on historical development or methodological grouped by similar topics; and its expected audience as research or professional community in the domain reviewed. This conceptual review aims the dual goal of knowledge integration and critique - on plan-driven and agile development life cycles for BPMS against two generic life cycles templates -; it is focused on practices – i.e. empirical professional development methodologies for BPMS -; it is realized from a non-neutral perspective – it aims to describe and compare the main identified plan-driven and agile development methodologies against a generic plan-driven BPMS life cycle and a generic agile Scrum-XP life cycle -; it uses a systematic selective coverage – it reviews only the plan-driven and agile development life cycles for BPMS found in a high-quality selective set of scientific publications for the 2000-2023 period-; its organization is methodological – it is uses a generic plan-driven and an agile development life cycle for BPMS as templates for the review -; and it is elaborated for a dual audience – academics doing research on BPMS and professionals developing BPMS -.

This SSLR method was carried out following the three main steps shown in Table 17 (Cooper, 1988; Paré et al. 2015; Templier and Paré 2015).

Table 17 - Systematic Selective Literature Review (SSLR) research method

Step	Purpose	Outcomes	Outcomes in this research
1) To formulate the research goal.	To state the expected research goal indicating the theoretical, or practical or both ones expected contributions.	• Research goal statement.	• To contribute to the literature with a conceptual descriptive-comparative review of two – one classic and one lightweight type - relevant development methodologies for BDAS, and provide to the practice useful recommendations regarding both development methodologies.
2) To define data sources and selective criteria.	To identify and agree the set of data sources to collect the studies, as well as to define the selection criteria.	• List of data sources. • Selection criteria statements.	• The two development methodologies for BDAS were selected according to the next criteria: 1) to select the classic methodology most cited in the literature; and 2) to select a modern and complete - i.e. it includes roles, phases, activities, and artifacts -lightweight development methodology reported from 2015-2022 period.
3) To collect studies.	To get the studies.	• Set of selected studies.	• To methodologies were identified, and theirs published references [13,14] were obtained.
4) To review and synthetize the findings from the collected studies.	To conduct the analysis and integration of finding.	• Structured schema of findings.	• We elaborated a generic lightweight development methodology using the ISO/IEC 29110 basic profile standard.
5) To elaborate report of findings.	To produce visible results.	• Research results.	• This chapters was elaborated.

Tables 18 reports the set of the 8 studies on plan-driven or agile development life cycles for BPMS found in the set of the 25 selected high-quality scientific publications from the disciplines of Information Systems (18 publications), and Software Engineering (7 publications). Table 18 reports type of development life cycle between plan-driven or agile, publication domain, publication name, type of publication – JCR journal or book-, publication impact factor if available, publication year, study title, and number of citations. Table 18 reports firstly the 4 plan-driven studies and secondly the 4 agile ones.

Table 18 - Set of 8 studies on Plan-Driven and Agile Development Life Cycles for PAIS/BPMS

Type of PAIS/|BPMS Life Cycle	Publication Domain	Publication Name	Type of Publication	Publication IF	Publication Year	Study Title	Citations
Heavyweight	Information Systems	CACM	JCR journal		2007	Business process development life cycle methodology.	114
Heavyweight	Software Engineering	IST	JCR journal		2008	A methodology for business process improvement and IS development.	103
Heavyweight	Information Systems	EIS	JCR journal		2015	A methodology proposal for collaborative business process elaboration using a model-driven approach.	19
Heavyweight	Information Systems	BPMJ	JCR journal		2020	Applications of business governance and the Unified BPM Cycle in public credit recovery activities.	4
Agile	Information Systems	BPM CONF	Book (Conference Proceedings)		2013	An agile BPM project methodology.	63
Agile	Information Systems	BPM HANDBOOK	Book		2015	Applying agile principles to BPM.	18
Agile	Information Systems	ISEBM	JCR journal		2016	Agile business process development: why, how and when—applying Nonaka's theory of knowledge transformation to business process development.	81
Agile	Information Systems	PROCEDIA	Book (Conference Proceedings)		2017	An agile business process improvement methodology.	50

3.1.3.4 NON-AGILE BPMS/POIS SOFTWARE DEVELOPMENT METHODOLOGIES*

It is important to analyze plan-driven software development life cycles (methodologies) for BPMS so that We can have a good perspective of what was doing in the BPMS*/PAIS field. To find methodologies that can help this research we found three that are:

- M1: Business Process Development Life Cycle Methodology (Papazoglou & van den Heuvel, 2007).
- M2: A methodology for business process improvement and IS development (Damij et al., 2008).
- M3: A methodology proposal for collaborative business process elaboration using a model-driven approach (Mu et al., 2015).
- M4: Applications of business governance and the Unified BPM Cycle in public credit recovery activities (Nascimento et al., 2019).

M1 by Papazoglou and van den Heuvel (2007) (represented in Figure 22) describes every single phase, there is a lack of information in some activities, the artifacts are not reported and should be deduced by the reader, the roles are not clearly defined and it does not describe what activities must be implemented by any role. The classic style of this methodology denotes a lot of extra work that can be not acceptable in an agile environment. This methodology was not implemented in a real-world application. The Phases for this methodology are Planning, Services and Process Analysis and Design, Construction and Testing, Provisioning, Deployment, and Execution and Monitoring. This methodology could be implemented easily by people who are used to working with waterfall.

M2 by Damij et al. (2008) is simply called TAD methodology that contains six phases: Business process identification, Business process modeling, Business process improvement, Object model development, Design, and Implementation. This methodology has its processes and artifacts very well defined. The authors specified in the article all the steps that should be followed in case you want to implement it. The extra work that needs to be implemented with this methodology could be very heavy. The authors provide an example of how to implement the methodology but requires a deeper knowledge of the business processes.

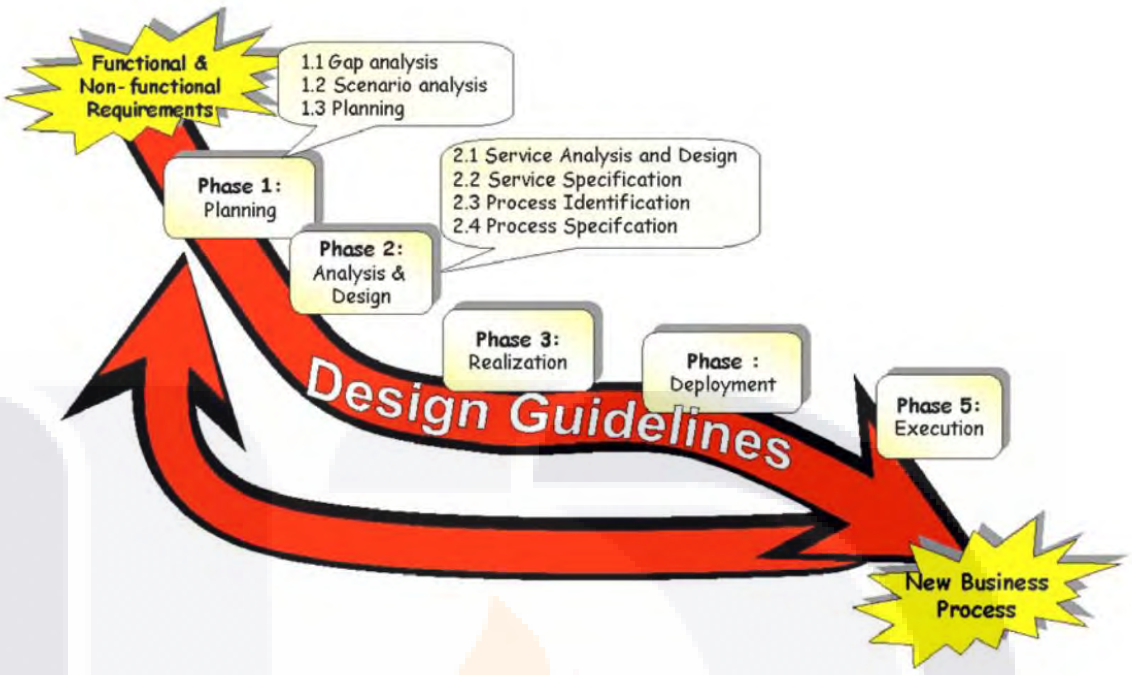


Figure 22 - Business Process Development Life Cycle Methodology Roadmap (Papazoglou & van den Heuvel, 2007)

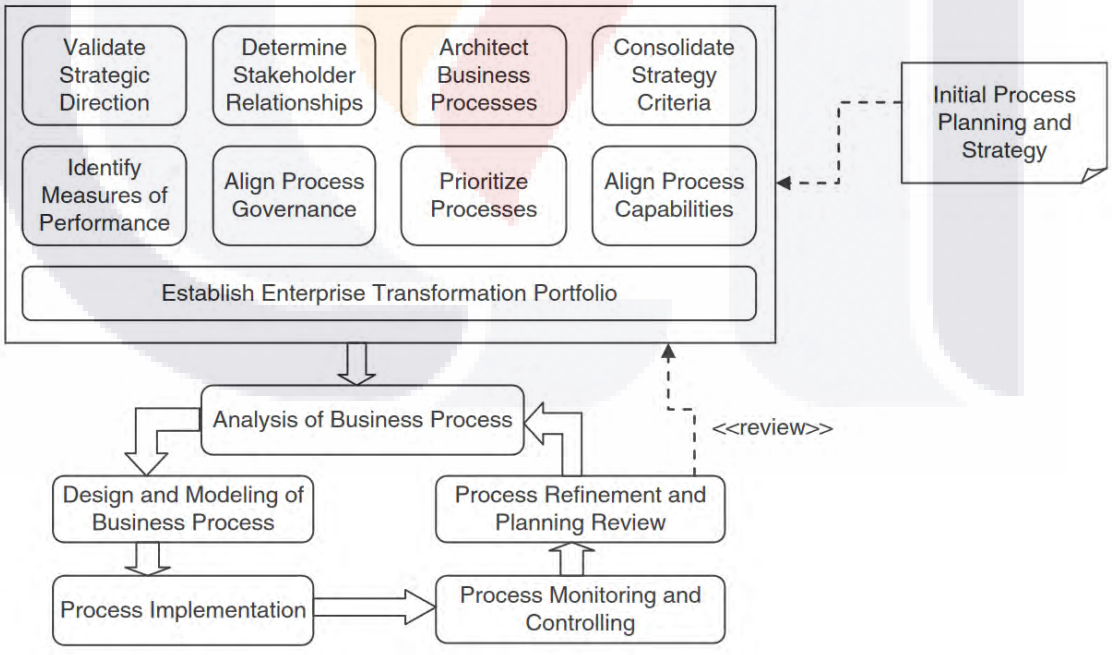


Figure 23 - BPM framework (Macedo de Morais et al., 2014)

M3 by Mu et al. (2015) defines four phases: Organizational, Functional, Informational, and Process. It also defines a Collaborative Meta-Model which represents the roles and their functions. The organizational phase defines the roles, the relationships between the partners, and the objectives of the main network. In the Functional phase the tasks are defined defining the inputs and the outputs. The Information phase is where all messages are modeled and transferred between functions. The Process phase is where the BPM is modeled with all their characteristics and relationships between partners. Finally, the Collaborative Meta-Model is divided into four packages that divide the vision in views for organizational, functional, informational, and process. This methodology has a deeper description of all the phases, processes, and artifacts. It is important to have a very strong knowledge of UML and BPN to create all the needed documentation across all phases. Figure 24 represents the methodology created by Mu et al. (2015).

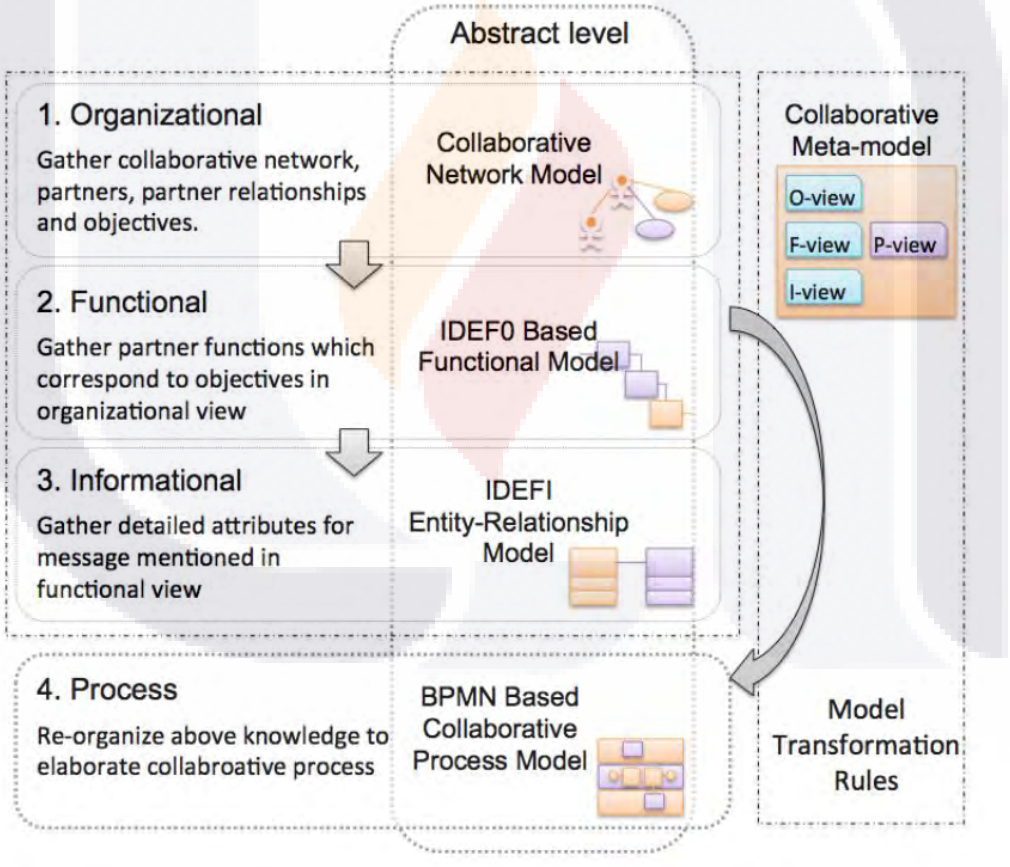


Figure 24 - Collaborative process elaboration methodology (Mu et al., 2015).

M4 by Nascimento et al. (2019) uses Governance Structure for preparatory stages to implement BPM, uses information from PMBOK and NBR ISO 10013 to implement the BPM actions following best practices, and finally uses the Unified BPM Cycle by Baldam et al. (2014) with its four phases: Planning, Analysis and Modeling, Implementation, and Monitoring. Figure 10 displays all those phases and how they interact. Governance Structure is the phase where all the organization structure is understood. The second phase uses the PMBOK and NBR ISO 10013 guidelines to define the BPM processes. The planning phase prioritizes the activities to be done by the team. Analysis and modeling phases understand the current state of the BPM and create possible improvements. The implementation phase is where all the actions are implemented. Monitoring phases are where all done activities are under visualization to ensure the quality of the implemented activities. The authors implemented the methodology in a real-life example even though there is not a deep explanation of every single phase and the information about roles, activities or artifacts is not available. Figure 25 represents this methodology.

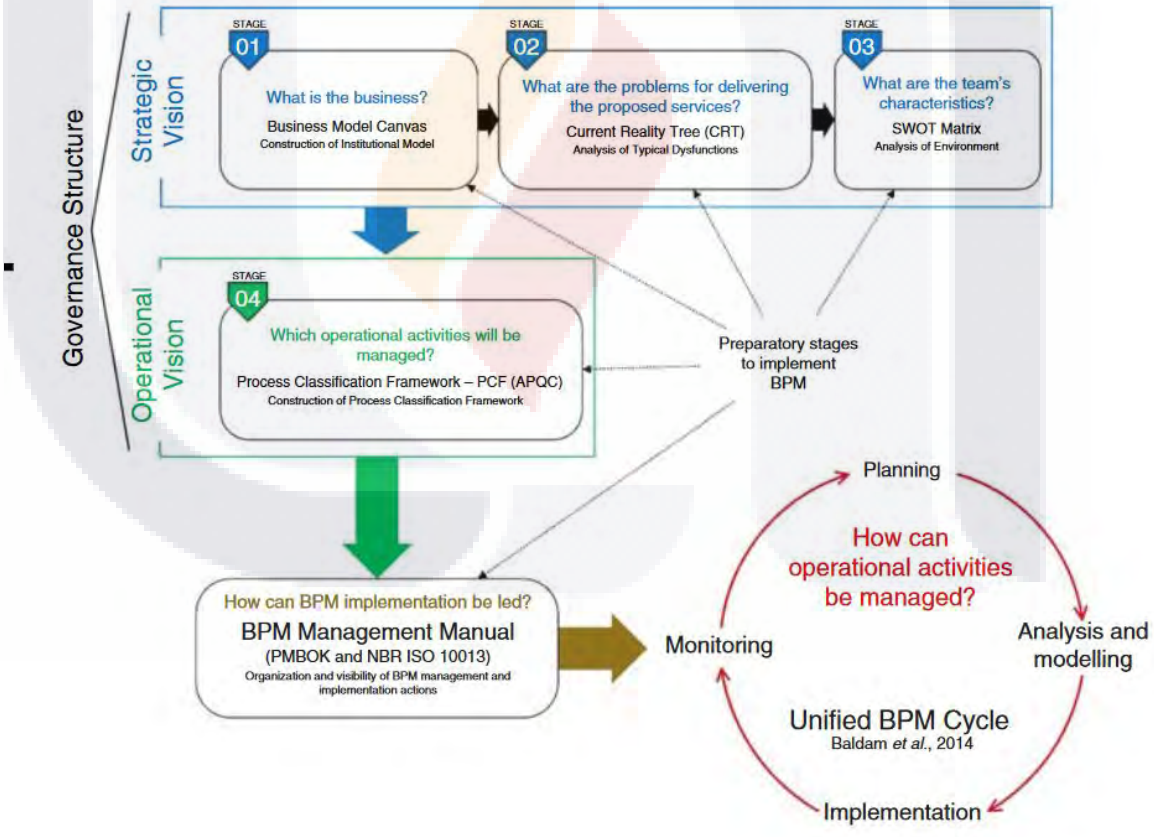


Figure 25 - Structure of corporate governance (Nascimento et al., 2019).

Dumas *et al*. (2018) defined the PAIS life cycle with six phases: Process Identification, Process Discovery, Process Analysis, Process Redesign, Process Implementation, and Process Monitoring. Process Identification is where the business problem is addressed, the outcome is a new or updated business process. Process Discovery is the phase where the process is documented. Process Analysis is where all the issues are discovered and prioritized in a list to resolve them. Process Redesign is where the processes are improved to resolve issues and fulfill the desired performance. The Process Implementation phase is where the issues are resolved based on the discovery phase. Process Monitoring is the phase where the redesigned processes are measured to find out if the desired performance is met.

The BPM lifecycle by Dumas *et al*. (2018) is represented in Figure 26, it helps us to understand what is the role of technology in BPM and is a key instrument to improve business processes.

To obtain the rationale and related studies of the relevance, need and most important plan-driven and agile development life cycles for BPMS, a Systematic Selective Literature Review (SSLR) research method was conducted. SSLR method is a research method in the descriptive-interpretative research approach, i.e., literature review, uses bibliographic research methods and conceptual analysis (Cooper 1988; Glass et al. 2004; Paré et al. 2015; Templier and Paré 2015).

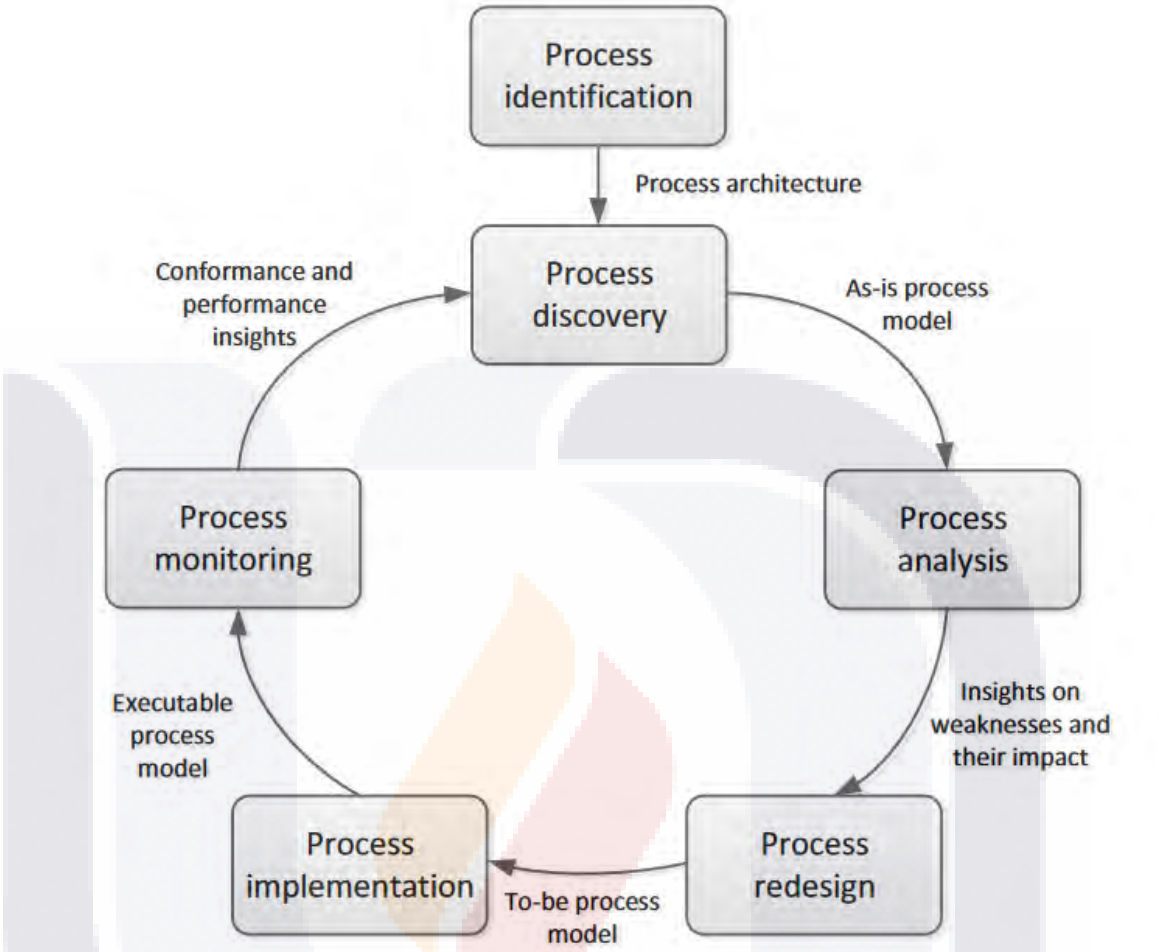


Figure 26 - The BPM lifecycle (Dumas et al., 2018).

Table 29 is an overview of the The BPM lifecycle detailing all Roles. Activities, and Artifacts reported by Dumas (2018).

To compare those methodologies we can use the BPM lifecycle (Dumas et al., 2018) to match their phases represented in Table 20

Table 19 - The BPM lifecycle detailed.

PHASE-WORKFLOW COMPONENTS CATEGORIES	The BPM lifecycle (Dumas et al., 2018). PHASE-WORKFLOW COMPONENT DESCRIPTION
Roles	1. Management Team 2. Process Owners 3. Process Participants 4. Process Analysts 5. Process Methodologist 6. System Engineers 7. BPM Group
Activities-Tasks	**Process Identification:** "Process identification refers to those management activities that aim to systematically define the set of business processes of an organization and establish clear criteria for selecting specific processes for improvement. The output of process identification is a process architecture, which represents the processes and their interrelations." (Dumas et al., 2018, p. 35). **Activities:** {1. Process architecture definition. 2. Process selection}
	Process Discovery: "The current state of each of the relevant processes is documented, typically in the form of one or several as-is process models." (Dumas et al., 2018, p. 22). **Activities:** {1. Defining the Setting. 2. Gathering the Required Information. 3. Modeling the Process. 4. Assuring Model Quality}
	Quantitative Process Analysis: "There is not a single way of producing a good process analysis, but rather a range of principles and techniques that tell us which practices typically lead to a “good” process analysis". (Dumas et al., 2018, p. 35). **Activities**: {1.Value-Added Analysis. 2. Waste Analysis. 3. Stakeholder Analysis and Issue Documentation. 4. Root Cause Analysis. }
	Qualitative Process Analysis: "Qualitative analysis is a valuable tool to gain systematic insights into a process. However, the results obtained from qualitative analysis are sometimes not detailed enough to provide a solid basis for decision making". (Dumas et al., 2018, p. 255). **Activities**: {1.Flow Analysis. 2. Queues 3. Simulation. 4. }
	Process Redesign: "The thorough analysis of a business process may lead to the identification of a range of issues. For example, bottlenecks slow down the process or the cost of process execution is too high". (Dumas et al., 2018, p. 297). **Activities**: {1. Transactional Methods. 2. Transformational Methods. }
	Process Implementation: "Conceptual process models must be systematically reworked into executable process models to be interpreted and automatically executed by a software system, such as a BPMS.". (Dumas et al., 2018, p. 371). **Activities:** {1. Identify the automation boundaries. 2. Review manual tasks. Complete the process model. 3. Bring the process model to an adequate level of granularity. 4.Specify execution properties}
	Process Monitoring: "Process monitoring is about using the data generated by the execution of a business process in order to extract insights about the actual performance of the process and to verify its conformance with respect to norms, policies, or regulations.". (Dumas et al., 2018, p. 371). **Activities:** {1. Offline Process Monitoring. 2.Online Process Monitoring}
Artifacts	**Process Identification:** {1. Process Architecture}
	Process Discovery: {1. Business Process modeled in BPMN}
	Qualitative Process Analysis: {1. Classification of Steps tables. 2.- Issue Register Documents. 3. Pareto Charts. 4. PICK Charts. 5. Cause-Effect Diagrams. 6. Why-Why Diagrams. }

	Quantitative Process Analysis: {1. Cycle Times Tables. 2.- Processing Times Tables. 3. Task Cycle Times Tables. 4. Analysis of Cycle Times. 5. Cost Calculation Tables. 6. Histograms Simulation Charts.}
	Process Redesign: {1. Devil's Quadrangle. 2. The Process Model Canvas. 3. Product Data Model}
	Process Implementation: {1. Executable Models with BPMS}
	Process Monitoring: {1. Operational Dashboard. Tactical Dashboards. 2. Strategic Dashboards. 3. Event Logs. 4. Dependency Graphs. 5.Dotted Chart. 6. Timeline Chart. }

Table 20 - Non-Agile BPMS Methodologies compared.

PHASE-WORKFLOW COMPONENTS CATEGORIES	Business Process Development Life Cycle Methodology (Papazoglou & van den Heuvel, 2007). PHASE-WORKFLOW COMPONENT DESCRIPTION	A methodology for business process improvement and IS development (Damij et al., 2008). PHASE-WORKFLOW COMPONENT DESCRIPTION	A methodology proposal for collaborative business process elaboration using a model-driven approach. (Mu et al., 2015). PHASE-WORKFLOW COMPONENT DESCRIPTION	Applications of business governance and the Unified BPM Cycle in public credit recovery activities. (Nascimento et al., 2019). PHASE-WORKFLOW COMPONENT DESCRIPTION
Roles	No reported	No reported	No reported	No reported
Activities-Tasks	**Planning Phase:** "Planning sets the scene for all ensuing phases by analyzing the business case of all viable mixtures of development approaches and realization strategies". (Papazoglou & van den Heuvel, 2007, p.4). **Activities**: {1. Gap Analysis, 2.Scenario Analysis, 3. Planning} **Service and Process Analysis and Design:** "Service analysis aims at identifying, conceptualizing and rationalizing business processes as a set of interacting Web services". . **Activities**: {1. Service Analysis and Design, 2. Service Specification, 3. Identifying Processes, 4. Specifying Processes}	**Business process identification:** "The first phase deals with identifying the business processes of the enterprise discussed. To do that, we have to conduct interviews with the management at different levels". (Damij et al., 2008, p.1130). **Activities:** {1. Business processes, 2. Work processes, 3. Process table} **Business process modelling:** "Most of problems faced by enterprises concern internal business procedures that are neither well defined nor particularly efficient. The business process modelling system is a computer-based, potential solution to these problems. It is a system for managing a series of tasks (actions) defined for one or more procedures". (Damij et al., 2008, p.1131). **Activities:** {1. Create Activity table, 2. Create Property table.}	**Organizational:** "Organizational modeling is not a new subject in enterprise modeling. But most of the organizational models only define the organization chart of enterprises, in terms of responsibilities, departments and workers. In a collaborative situation, the structure is a graph (in discrete mathematics terms) rather than a tree". (Mu et al., 2015, p.7). **Activities**: {1. Building a collaborative network model} **Functional:** "The requirements for the functional model are to obtain partner functions, to simplify user modeling tasks and to decrease user workload. The functional model only collects functions that partners want to share and which can be published to other partners". (Mu et al., 2015, p.10). **Activities: {**1.Create IDEF1 Model Unit.**}**	**Governance Structure:** "To achieve the proposed objectives, preliminary consultations were required for all available collections of official documents (laws, ordinances, instructions and dispatches of administrative-managerial content) because they were a very rich and stable source of data". (Nascimento et al., 2019, p.316). **Activities:** {1. Construction of Institutional Model, 2. Analysis of Typical Dysfunctions, 3. Analysis of Environment.} **BPM Management Manual: U**ses the PMBOK and NBR ISO 10013 guidelines to define the BPM processes. **Activities:** {1. Organization and visibility of BPM management, 2. Kmplementation actions.}

	Realization: "Once the service- and process specifications have reached a steady state, they need to be transformed into service implementations". (Papazoglou & van den Heuvel, 2007, p.9). **Activities**: {1. Code Web Services, Code Business Processes}	**Business process improvement:** "The relationship between the essence of business process modelling and overall business effectiveness and the efficiency of the organization depends on the consumer's satisfaction with the desired output". (Damij et al., 2008, p.1136)**.** **Activities:** {1. Process analysis. 2. Process simulation.}	**Informational:** "The basic need for the informational model of MISE 2.0 is to model messages, which are transferred among business functions, and to model the properties of each message, which are reused in the BPEL transformation. IDEF1 [22] and UML class diagrams are both suitable for modeling informational". (Mu et al., 2015, p.12). **Activities:** {1. Model Messages. }	**Planning:** Prioritizes the activities to be done by the team. **Activities:** {1. Create BPM Management Manual.}
	Deployment: In this phase the new services are deployed. **Activities:** {1. Publish the service interface. 2. Deploy the Web service and business process. 3. Publish service implementation details}	**Object model development:** "This model is developed using the information collected in the tables, particularly the property table". (Damij et al., 2008, p.1138). **Activities**: {1. Initial object model. 2. Final object model}	**Process**: "In the process modeling domain, a number of models have been defined, such as flow charts IDEF3, Petri nets, Event Process Chains of ARIS, activity diagrams of UML and, more recently, BPMN". (Mu et al., 2015, p.10). **Activities:** {1. Create BPMN Models}	**Analysis and modelling:** Understand the current state of the BPM and create possible improvements. **Activities:** {1. Evaluate}
	Excecution: "During the execution phase, the business processes and supporting Web services are fully deploye and made operational**".** (Papazoglou & van den Heuvel, 2007, p.9). **Activities**:	**Design:** "Deals with designing the system and preparing it for implementation". (Damij et al., 2008, p.1140). **Activities:** {1. Operations, Design model, 3. Algorithms}		**Implementation:** Is where all the actions are implemented. **Activities:** {1. Implementation of audit activities, 2. Information collection and verification, 3. Management of findings, 4. Preparation of conclusions. }
		Implementation: "Deals with the implementation of the models developed in the previous phases. The inputs to the implementation phase are the object model and design model". (Damij et al., 2008, p.1141). **Activities:** {}		**Monitoring:** is where all done activities are under visualization to ensure the quality of the implemented activities. **Activities:** {1. Monitoring}

Artifacts	**Planning Phase**: No reported **Service and Process Analysis and Design**: No reported **Realization**: No reported **Deployment**: No reported **Excecution**: No reported	**Business process identification**: {1. List of strategic goals, 2. List of business processes, 3. Organizational scheme of the enterprise, 4. plan of interviews with management at operational level } **Business process modelling**: {1. Activity table, 2. Property table, 3. Flowchart} **Business process improvement**: {1.Object model} **Object model development**: No reported **Design**: {1. Design Model} **Implementation**: {1. Program Codes}	**Organizational**: {1. Collaborative network model} **Functional**: {1. Functional Model} **Informational**: {1. Informational Model} **Process**: {1.BPMN Models}	**Governance Structure**: {1. Canvas Business Model, 2. Current Reality Tree, 3. SWOT Analysis, 4. Process Classification Structure} **BPM Management Manual**: {1. PMBOK-PMI, 2. NBR ISO 10013.} **Planning**: {1. BPM Management Manual} **Analysis and modelling**: No reported **Implementation**: No reported **Monitoring**: No reported

3.1.3.5 AGILE BPMS/POIS SOFWARE DEVELOPMENT METHODOLOGIES*

Agility is needed for BPM to deal with challenges and able to deal with process change (Badakhshan et al., 2019). "***Emerging technologies in BPM, such as process mining, machine learning, and the Internet of Things (IoT), enable organizations to evaluate processes on a real-time basis through real-time connectivity, so process criteria like time, quality, and cost can be evaluated on an ongoing basis. Modern technologies help organizations to identify and prioritize processes rapidly, initiate necessary process changes, and manage process models timely***" (Badakhshan et al., 2019, p. 9).

To be considered agile a process needs to have Flexibility, Leanness, and Continuity (Badakhshan et al., 2019). Figure 27 displays the Agile BPM Framework created by Badakhshan *et al*. (2019) propose three columns:

- Column A: Talks about how an organization should be ready for process change.
- Column B: None of the activities should decrease the quality, economy, and simplicity perceived by customers.
- Column C: Is focused on how an organization should be always scouting for new trends like technologies that can enable agility in BPM.

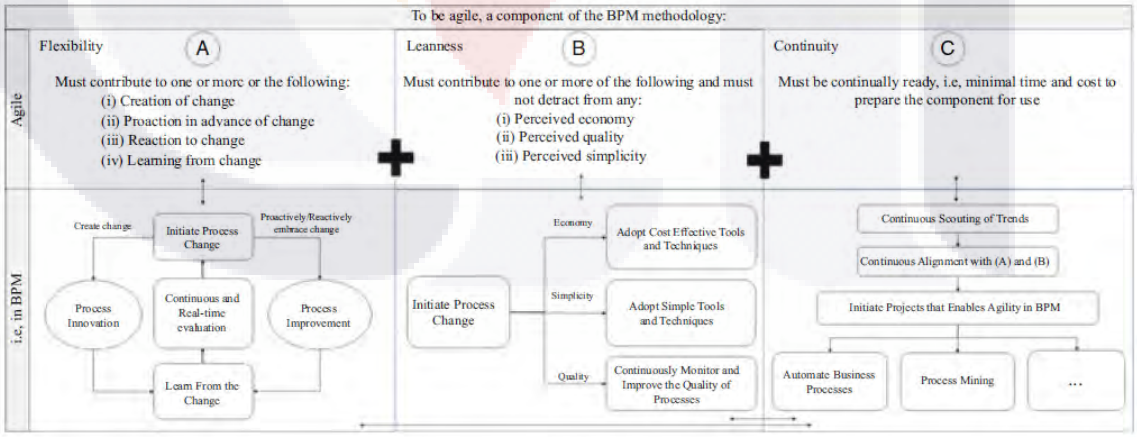


Figure 27 - Agile BPM framework

Through the literature, there are some agile methodologies for BPM that try to create business processes in an agile way. Silva *et al.* proposed AGILIPO (2009) that taking into account that all business processes are incomplete and need to be

constantly changed, Rachid Meziani and Rodrigo Magalhães (2009) created a complementary agile methodology for AGILIPO with five complementary steps, Ventura and Zacarias (2017) created an agile methodology for improving Business Processes based on daily practices with iterative processes and people involved. All those methodologies can be implemented for working in Business Processes with agility.

This Ph.D. dissertation is focused on methodologies for developing information systems using a BMPS with low-code platforms.

For this section, we found two Agile BPMS*/PAIS methodologies that can be applied for developing an information system using a low-code development platform:

- M1: An agile BPM project methodology (Thiemich & Puhlmann, 2013).
- M2: Applying Agile Principles to BPM (Rosing & Gill, 2015).
- M3: Agile business process development: why, how and when—applying Nonaka's theory of knowledge transformation to business process development (Bider & Jalali, 2016).
- M4: An agile business process improvement methodology (Martins & Zacarias, 2017).

M1 by Thiemich and Puhlmann (2013) created an Agile BPM Project Methodology, shown in Figure 28, using the IBPM Methodology, a traditional waterfall methodology, and the Scrum framework trying to provide sustainable and continuous improvements. The Agile BPM Project Methodology defines Activities, Methods, and Artifacts considering Pregame phases such as Scoping, Kick-Off, and Sprint 0. The Game phase is covered by Sprint 1-n while Postgame Phase has the Releasesprint. If this methodology merges IBPM Methodology some elements are necessary to know to implement that can add more difficult to be implemented. There are some gaps in the documentation some activities, artifacts, and methods are not clearly described so the users can be implementing in the wrong way the methodology. Agile BPM Project Methodology was tested with a real service portal project.

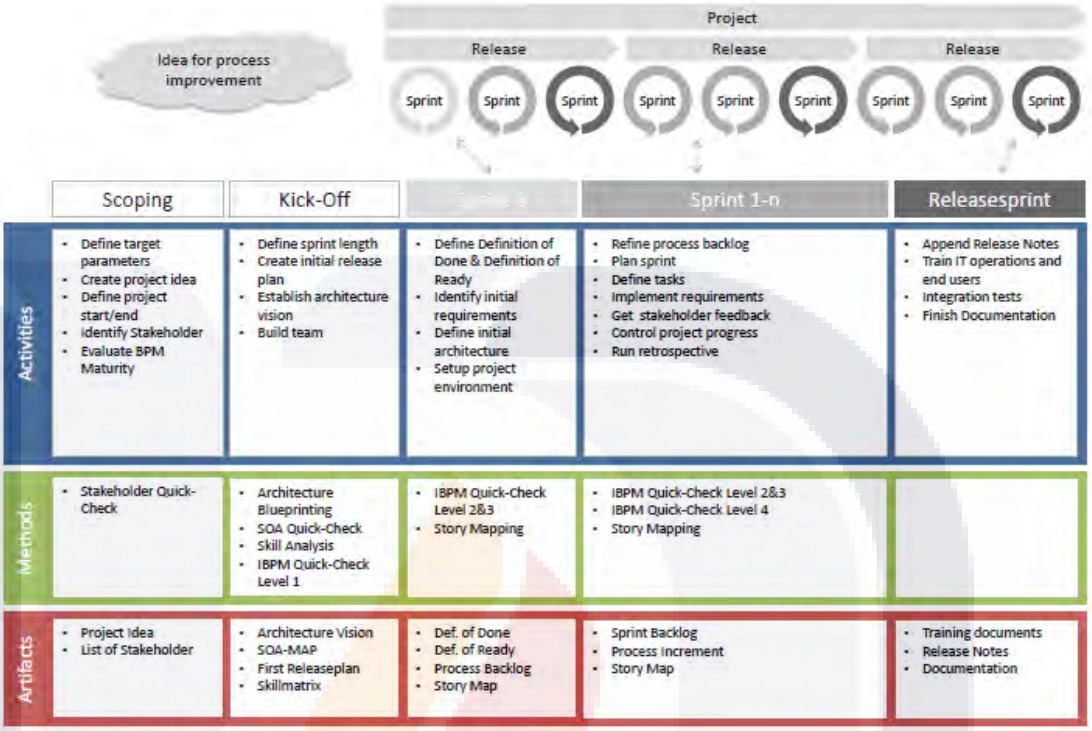


Figure 28 - Agile BPM Framework Overview (Thiemich and Puhlmann, 2013).

M2 by von Rosing *et al*. (2015) (see Figure 29) created an Agile BPM Methodology that has six phases such as Analyse, Plan, Design, Build, Test, and Deploy. All those phases contain a clear workflow of the processes with some common questions and actions that must be performed in every step. Most of the roles, events, artifacts, and methods are not documented so it is very easy to get lost trying to implement the methodology. This Agile BPM Methodology was not tested with a prototype or real-life problem.

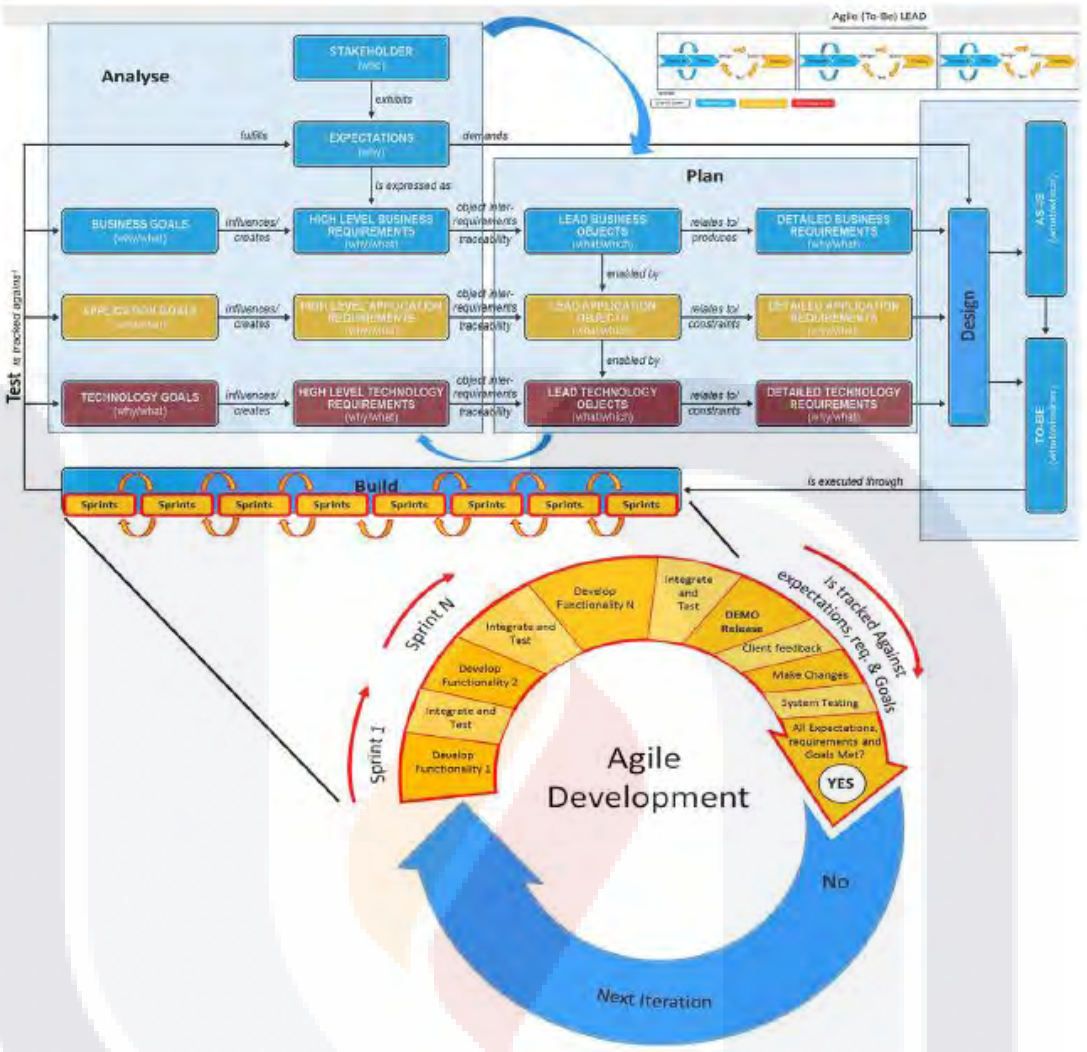


Figure 29 - Agile BPM Overview (von Rosing et al., 2015).

M3 by Bider and Jalali (2016) (see Figure 30) uses the SECI model created by Nonaka (1994) and practical knowledge for creating an agile methodology. This model has 3 phases: Socialization on this phase the stakeholders and the development team interact and share knowledge so that the development can cover their needs. Embedment in this phase the BPM specialist work with business people to create and implement tacit knowledge into real business processes. Adoption in this phase the BPM is running and the practitioners are gaining valuable knowledge for sharing for the next business process creationg or to adjust current processes. Figure 11 displays the SECI Model. This methodology was tested with real cases and its authors reported some advantages and disadvantages to using it. This methodology does not report Roles, Activities, and Artifacts clearly so it would be impractical to implement for any development.

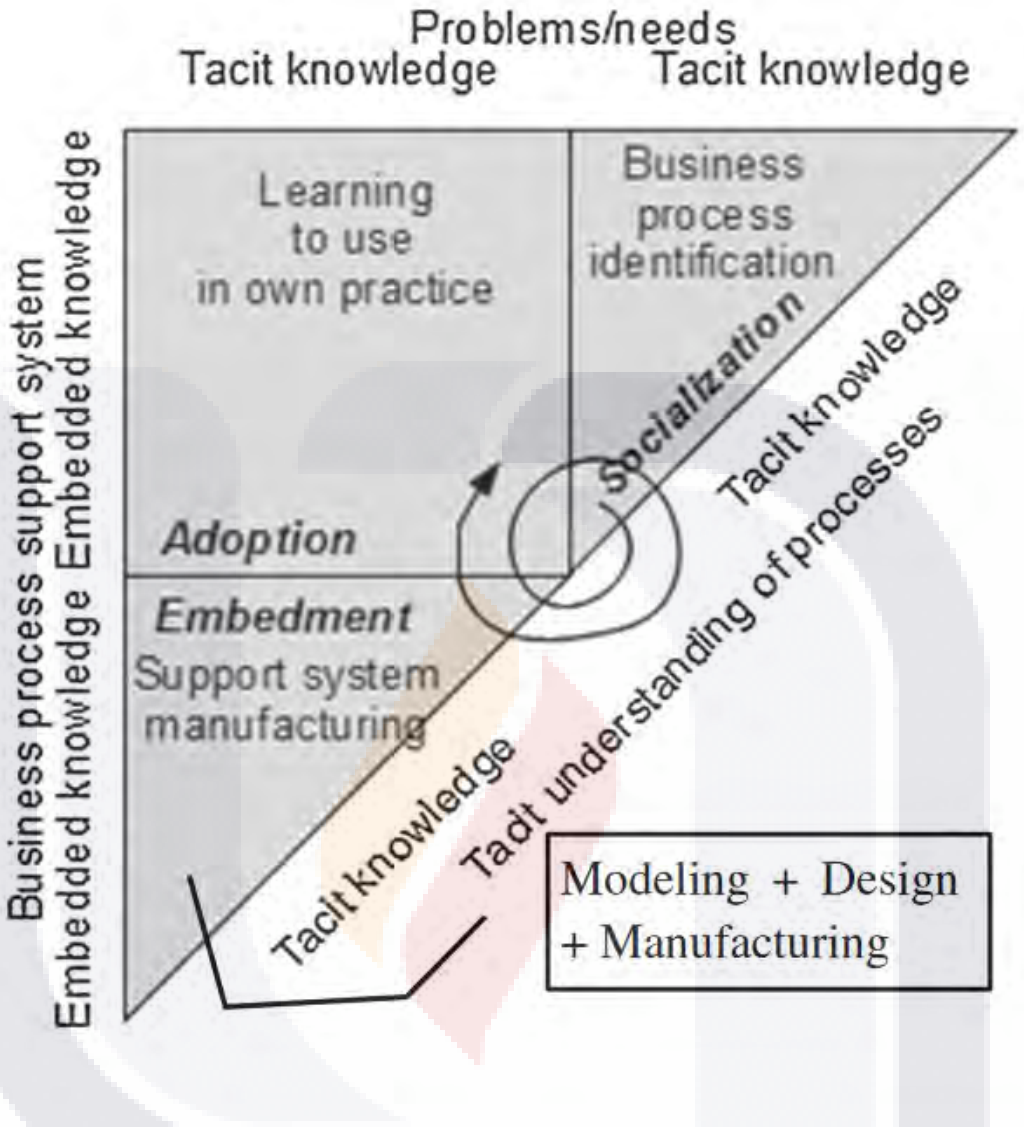


Figure 30 - SEA—knowledge transformation in the agile process development (Bider & Jalali, 2016).

M4 by Martins and Zacarias (2017) (see Figure 31) is a proposal from adopting traditional BPPAM Methodology into Agile to act quickly to changes from non-experts users. This methodology consists of three phases: Business Process Discovery (BPD), Business Process Supervision (BPS), and Business Process Assessment and Improvement (BPAI). Figure 31 represents the Agile BPPAM methodology.

Phase 1: Business Process Discovery (BPD): The main goal is developing an organizational profile to understand the business processes of a company. In this phase, we need to learn about the company and model its business processes. Phase 2: Business Process Supervision (BPS): Control mechanisms are created and make sure that stakeholder brings models to real business activities. Phase 3:

Business Process Assessment and Improvement (BPAI): In this phase, the company identifies its strengths, weaknesses, existing improvement activities, and key areas for improvement.

In this methodology, there are some activities mentioned for Phase 1 like Learning (Eliciting) Business and Modelling Business, for Phase 2 there are three activities for controlling mechanisms such as: comparing real business activities with base business models, annotating/reviewing models, and, identify new business descriptions. Phase 3 is where the Business Analyst implements corrections on current processes. There are few details for activities, roles, and artifacts so that could be very complicated to implement for a practitioner without detailed documentation.

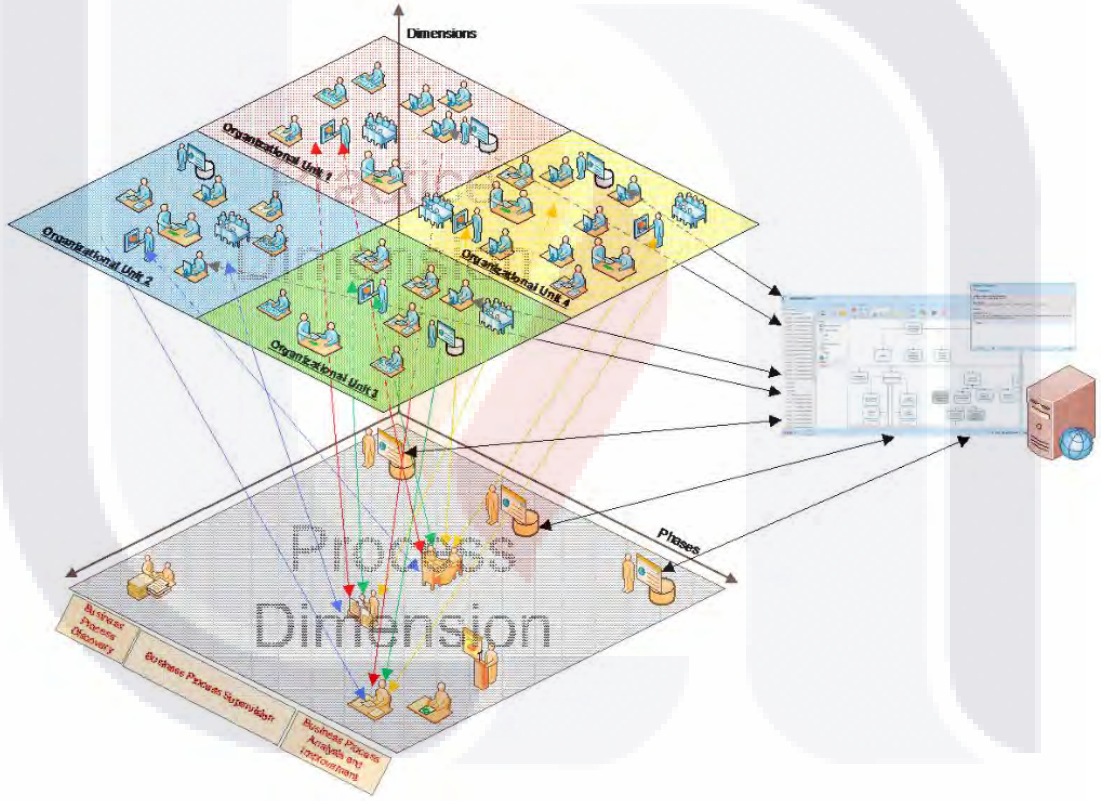


Figure 31 - Agile BPPAM methodology by Zacarias (2017).

Table 21 has all the information provided for the Agile BPMS Methodologies and is compared against Scrump- Xp.

Table 21 - Aguile BPMS Methodologies Comparative

PHASE-WORKFLOW COMPONENTS CATEGORIES	An Agile BPM Project Methodology (Thiemich & Puhlmann, 2013).	Applying Agile Principles to BPM (Rosing & Gill, 2015).	Agile business process development: why, how and when—applying Nonaka's theory of knowledge transformation to business process development (Bider & Jalali, 2016).	An agile business process improvement methodology (Martins & Zacarias, 2017).
Roles	1. BPM Process Owner 2. BPM Master 3. BPM Team	1. Process Owner 2. Agile coach 3. Cross-functional team	No Reported	No reported
Activities-Tasks	**Scoping:** In this phase is where the project is defined and the stakeholders are identified. **Activities:** {1. Define target parameters. 2. Create project idea. 3. Define project start/end. 4. Identify Stakeholder. 5. Evaluate BPM Maturity.}	**Agile Analysis:** "Agile analysis, in the context of Agile BPM, suggests active collaboration with the stakeholders to identify the requirements with necessary details at the release and iteration levels, instead of trying to get the complete detailed requirements up-front". (Rosing & Gill, 2015, p.564). **Activities**: {1. Expectations. 2. Business Goals. 3. Application Goals. 4. Technology Goals. 5. High Level Business Requirements. 6. High Level Application Requirements. 6. High Level Technology Requirements. }	**Socialization:** "The nature of the first phase consists in transferring tacit knowledge on the desired process from the stakeholders to the design team" (Bider & Jalali, 2016, p.17). **Activities**: No reported	**Business Process Discovery (BPD):** "Aims at developing an organisational profile in order to understand business processes which contain information about people, activities, technology and data" (Martins & Zacarias, 2017, p.133). **Activities:** {1. Learning (Eliciting) Business. 2. Modelling Business. }
	Kick-Off: Is where the initial parameters are set and the team is built. **Activities:** {1. Define sprint length. 2. Create initial release plan. 3. Establish architecture vision. 4. Build team.}	**Agile Planning:** "Traditional ways of BPM planning focus on the detailed up-front planning. Agile BPM ways of working require planning at project, release, iteration, and day level. Agile BPM focuses on initial high-level project plan that outlines number of project releases, resources, risks, and cost and benefits estimates". (Rosing & Gill, 2015, p.565). **Activities:** {1. Lead Business Objects. 2. Lead Application Objects. 3. Lead Technology Objects. 4. Detailed Business Requirements. 5. Detailed Application	**Embedment**: "In this cycle, process modeling, system design and manufacturing are merged into one phase Support system manufacturing (Embedment)" (Bider & Jalali, 2016, p.17). **Activities:** No reported	**Business Process Supervision (BPS):** "Formal control mechanisms are designed in order to ensure that operational stakeholder carried out real business activities as described by business models" (Martins & Zacarias, 2017, p.134). **Activities:** {1. Compare real business activities with base business models. 2. Annotate/review models. 3. identify new business descriptions. }

		Requirements. 6. Detailed Technology Requirements. }		
	Sprint 0: First sprint where is defined all the elements that are needed in a normal sprint. **Activities:** {1. Define Definition of Done & Definition of Ready. Identify initial requirements. 3. Define initial architecture. 4. Setup project environment}	**Agile Architecture and Design:** "Agile design for BPM can kick off by reviewing the existing As-Is process model and identified requirements for the target To-Be process model. Instead of a detailed up-front design, a high-level design for the To-Be process can be developed at the start of the project". (Rosing & Gill, 2015, p.566)**. Activities:** {1. To-Be. 2. As-Is. }	**Adoption**: "one big cycle is substituted by many smaller and shorter ones. The system is built iteratively starting with the basic functionality that does not limit flexibility of process participants to experiment with the new process. During the usage of the basic system, better understanding of the needs is acquired, which is converted in adding details to the system in the next cycle." (Bider & Jalali, 2016, p.17). **Activities:** No reported	**Business Process Assessment and Improvement (BPAI)**: "Is a mean for organisations to identify their strengths, weaknesses, existing improvement activities and key areas for improvement" (Martins & Zacarias, 2017, p.134). **Activities**: No reported
	Sprint 1-n: Iterative Process where an increment is built working providing value to the customer. **Activities:** {1. Refine process backlog. 2. Plan sprint. 3. Define tasks. 3. Implement requirements. 4. Get stakeholder feedback. 5. Control project progress. 6. Run retrospective.}	**Agile Build:** "Traditional ways of working focus on big-bang product or service development in the build phase. Agile ways of working focus on building the product or service minimum marketable or viable features in small iterations based on the just-in-time user stories or requirements". (Rosing & Gill, 2015, p.567). **Activities:** {1. Defining the Product Backlog. 2. Sprint Planning Meeting. 3. Defining the Sprint Backlog. 4. Interrogating and Testing. 5. Demo Release. 6. Client Feedback Meeting. 7. Retrospective. 8. Refactoring. 9. System Changes. 10. System Testing. }		

	Releasesprint: Sprint where the team is focused on releasing done work. **Activities:** {1. Append Release Notes. 2. Train IT operations and end users. 3. Integration tests. 4. Finish Documentation. }	**Agile Testing:** "Although traditional ways of working around testing first do the testing once the whole product or service is developed, agile ways of working focus on testing the product or service minimum marketable or viable features in small iterations while the development is in progress". (Rosing & Gill, 2015, p.567). **Activities:** {1. Deployment to production. }		
Artifacts	**Scoping:** {1. Project Idea. 2. List of Stakeholder}	**Agile Analysis**: No reported.	**Socialization:** No reported	**Business Process Discovery (BPD):** No reported
	Kick-Off: {1. Architecture Vision. 2. SOA-MAP. 3. First Releaseplan. Skillmatrix. }	**Agile Planning**: {1. Definition of "done". 2. Release plans. 3. Product Backlog. 4. User story. }	**Embedment**: No reported	**Business Process Supervision (BPS):** No reported
	Sprint 0: {1. Def. of Done. 2. Def. of Ready. 3. Process Backlog. 4. Story Map.}	**Agile Architecture and Design:** {1. ModelTo-Be. 2. Model As-Is. }	**Adoption**: No reported	**Business Process Assessment and Improvement (BPAI)**: No reported.
	Sprint 1-n: {1. Sprint Backlog. 2. Process Increment. 3. Story Map}	**Agile Build:** {1. System Changes. 2. System Testing. 3. Kanban board. 4. Burndown chart. 5. Burnup chart. 6. Defect trend. 7. Decision Point. }		
	Releasesprint: {1. Training documents. 2. Release Notes. 3. Documentation}	**Agile Testing**: {1. Working product. }		

3.2 ANALYSIS OF CONTRIBUTIONS AND LIMITATIONS

Table 22 displays the analysis from related studies with their characteristics and limitations.

Table 22 - Contributions and are of improvements

Area	Contributions	Opportunities of Improvement
Software Engineering	1. Is the basic set of tools that help to develop software in an orderly way. 2. Research in this area has helped to improve technology. 3. How to manage the resources for the software process (phases, activities, artifacts, and resources (including humans))	1. Lack of research with experience in the industry. 2. Methodologies for low-code platforms. 3. Gap between new technologies and research.
Agile Methodologies	1. Close work between clients and developers provides better results. 2. Iterative work with small releases provides value to the customer. 3. Software is developing faster	1. Sometimes the agile methodologies are not implemented as intended. 2. Official guides can be very vague and can leave many doubts to the practitioners. 3. Lack of limits on the project could generate chaos.
Business Process Management Systems	1. Develop software with low-code platforms by people with low programming knowledge 2. Improve development time versus traditional programing languages. 3. Improve development costs.	1. Lack of methodologies 2. Paid low-code platforms can be very expensive. 3. Open source low-code platforms can be difficult to learn.
Non-agile BPMS*/POIS Software Development Methodologies	1. Good practices for working with BPM projects. 2. Proven methodologies in real-life projects. 3. Some methodologies can be very simple to implement for people that have worked on waterfall methodology.	1. Lack of methodologies for low-code platforms. 2. Very heavy methodologies that can be only implemented by BPM experts. 3. Most of the methodologies lack detailed documentation.
Agile BPMS*/POIS Software Development Methodologies	1. Use Scrum as the core for implementing a new agile methodology focused on BPMS. 2. One methodology is well explained and can be implemented by anyone that knows Scrum. 3. Propose interesting activities and artifacts that worth to be considering in future methodology projects.	1. Lack of proven methodologies in real projects. 2. Lack of detailed documentation for practitioners about how to use the methodologies. 3. Lack of methodologies.

4 DEVELOPMENT OF THE SOLUTION

As mentioned in Chapter 2 this Ph.D. The dissertation uses mainly the Design Science Research Methodology (DSRM) (Peffers et al., 2007) which is detailed in Table 2.1 and is divided into the next steps:

1. DSRM step 1 - Design problem identification and motivation.
2. DSRM step 2 - Definition of the Design Objectives, Design Restrictions, Resign Approach, Design Theoretical Sources, and Design Components for the expected Artifact.
3. DSRM step 3 - Design and development of the artifact.
4. DSRM step 4 - Demonstration of the artifact (Proof of Concept).
5. DSRM step 5 - Evaluation of the artifact.
6. DSRM step 6 - Communication of research results.

4.1 DSRM STEP 1 – DESIGN PROBLEM IDENTIFICATION AND MOTIVATION

Chapter 1 of this document contains all the detailed information for Problem Identification and its Motivation.

4.2 DSRM STEP 2 – DEFINITION OF THE DESING OBJECTIVES, DESING APPROACH, DESIGN THEORETICAL SOURCES, AND DESING COMPONENTS FOR THE EXPECTED ARTIFACT: AGILE BPM METHODOLOGY

4.2.1 DEFINITION OF THE DESING OBJECTIVES

The expected Design Objectives (DOs) to be archived in this work are:

1. DO.1 The designed artifact provides an agile (i.e. responsive, flexible, speedy, lean, simple, lightweight, and fine-grain documented (Conboy, 2009), (Qumer & Henderson-Sellers, 2008)) workflow—i.e. a value stream—for designing, building, and implementing a new minimum viable Agile BPM Methodology.
2. DO.2 The designed artifact is useful, easy to use, and valuable (Galvan et al., 2021) for small companies, software developers, and IT practitioners.

3. DO.3 The designed artifact is fine-grain documented including the roles-set component, phases-activities set component, and artifacts-templates-set component.

4.2.2 DESIGN RESTRICTIONS

For Design Restrictions (DRs) we need to take into account parameters such as time, budget, theoretical sources, and available software. The DRs that were agreed are:

1. DR.1 The designed artifact must be composed of design building blocks from relevant design theoretical sources (DTSs).
2. DR.2 The artifact must be designed in a short-term period (at most 6 months) and under the assigned research budget.
3. DR.3 The designed artifact must be documented in an Electronic Process Guide.

4.2.3 DESIGN THEORITICAL SOURCES

The Design Theoretical sources (DTSs) are the key sources of the design components that will be chosen to create the artifact. The DTSs selected were proposed based on the theoretical background and having reviewed the eight BPMS methodologies.

1. DTS.1 The BPM Lifecycle (Dumas et al., 2018)
2. DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)
3. DTS.3 APBM (Thiemich & Puhlmann, 2013).
4. DTS.4 ABPM (Rosing & Gill, 2015).

Every single element such as Roles, Activities, and Artifacts for the DTS will be considered and discussed with the team to get the Desing Components.

4.2.4 DESIGN COMPONENTS FOR THE EXPECTED ARTIFACT

Evaluating very carefully the DTS, we have selected the potential design components (DCS) to be used in the design of the artifact. Some components may be not used in the final design.

Table 23, Table 24, Table 25, and Table 26 have all the Design Components selected from the four DTS by the research team based on their experience and

expertise. An iterative process is going to be performed in order to get de most important components to design the artifact.

Table 23 - DTS.1 Theoretical rigorous SDLC for BPMS (Dumas et al., 2018)

Design Component	Design theoretical source (DTS)	Specific elements of the design component (DC) potentially to be used in the designed artifact
DC.1 The BPM Lifecycle Phases	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	{Process Identification, Process Discovery}
DC.2 The BPM Lifecycle Activities	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	{Process Identification [Process architecture definition, Process selection], Process Discovery [Defining the setting, Gathering the required information, Modeling the process, Assuring model quality]}
DC.3 The BPM Lifecycle Artifacts	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	{Process Identification [Process architecture of the selected process], Process Discovery [As-is business process model]}

Table 24 - DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)

Design Component	Design theoretical source (DTS)	Specific elements of the design component (DC) potentially to be used in the designed artifact
DC.4 Scrum-XP Roles	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	{Customer-Product Owner; Coach-Master; Development Team}
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	{Exploration, Product Planning, Iteration-Sprint Planning, Iteration-Sprint, Product Release}
DC.6 Scrum-XP Activities	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	{Exploration [product vision definition; product backlog (user story set) definition; product backlog (user story set) prioritization; optional: spike testing]} {Product Planning [product backlog (user story set) effort estimation; product backlog (user story set) negotiation; optional: style codifying standard definition]} {Iteration-Sprint Planning [iteration-sprint user story selection; iterationsprint user story task planning iteration-sprint user story plan negotiation]} {Iteration-Sprint [stand-up meeting; customer functional tests elaboration; simple design; codification and unit testing; increment integration and customer functional testing; iteration-sprint review and retrospective]} {Product Release [product releasing]}

DC.7 Scrum-XP Artifacts	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	{Exploration [product vision; product backlog]} {Product Planning [product backlog plan]} {Iteration-Sprint Planning [iteration-sprint plan]} {Iteration-Sprint [iteration-sprint Kanban board; iteration-sprint burndown chart; customer functional tests; simple architecture design; unit tests; unit codes; built increment; iteration-sprint agreements]} {Product Release [product done]}

Table 25 - DTS.3 APBPM (Thiemich and Puhlmann, 2013)

Design Component	**Design theoretical source (DTS)**	**Specific elements of the design component (DC) potentially to be used in the designed artifact**
DC.8 APBPM Phases	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	{Project Scoping, Project Kick-Off, Sprint 0, Sprint 1-n, Release Sprint}
DC.9 APBPM Activities	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	{Project Scoping [Define target parameters, Create project idea, Define project start/end, Identify Stakeholder, Evaluate BPM Maturity], Project Kick-Off [Define sprint length, Create initial release plan, Establish architecture vision, Build team], Sprint 0 [Define Definition of Done & Definition of Ready, Identify initial requirements. Define initial architecture, Setup project environment], Sprint 1-n [Refine process backlog, Plan sprint, Define tasks, Implement requirements, Get stakeholder feedback. Control project progress, Run retrospective], Release Sprint [Append Release Notes, Train IT operations and end users, Integration tests, Finish Documentation.]}
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	{Project Scoping [Project Idea, List of Stakeholder], Project Kick-Off [Architecture Vision, SOA-MAP, First Release plan, Skillmatrix], Sprint 0 [Def. of Done, Def. of Ready, Process Backlog, Story Map], Sprint 1-n [Sprint Backlog, Process Increment, Story Map], Release Sprint [Training documents, Release Notes, Documentation]}

Table 26 - DTS.4 ABPM (Rosing and Gill, 2015).

Design Component	Design theoretical source (DTS)	Specific elements of the design component (DC) potentially to be used in the designed artifact
DC.11 ABPM Phases	DTS.4 ABPM (Rosing and Gill, 2015)	{Agile Analysis, Agile Planning, Agile build, testing, and deployment, Agile build, testing, and deployment}
DC.12 ABPM Activities	DTS.4 ABPM (Rosing and Gill, 2015)	{Agile Analysis [High Level Business Requirements], Agile Planning [High-level project plan], Agile build, testing, and deployment [Defining the Sprint Backlog, Sprint Planning, Performing Sprint, Testing, Demo Increment, Client Feedback Meeting, Retrospective, Deploying Increment]}
DC.13 ABPM Artifacts	DTS.4 ABPM (Rosing and Gill, 2015)	{Agile Analysis [Selected business process and sub-processes, High-level user stories, Table of priorities and estimations], Agile Planning [Project plan], Agile build, testing, and deployment, Agile build, testing, and deployment [Sprint Backlog, Sprint Task Plan, Tests, Increment, Integrated Release]}

4.3 DSRM STEP 3 – DESING AND DEVELOPMENT OF THE ARTIFACT

To design the BPMS Methodology the research team applied the Means-Ends Analysis heuristic (Newell & Simon, 1972) (Greeno et All,1987) in four steps:

- Step 1. To represent the design problem defining an initial state S i, a pursued final state S f, a set of heuristic operators {HOx(Sy, Sz), …} that can transform the state Sy to the state S z, a set of design objectives {DOj, …} and design restrictions {DR k, …} expected to be satisfied by the final state S f, and two qualitative functions EvalDOs(DO's) and EvalDRs(DR's) to evaluate the logical satisfaction of DO's and DR's.
- Step 2. To set up the initial state S i and the pursued final state S f, and determine the initial qualitative evaluations EvalDOs(DO's) and EvalDRs(DR's) for the initial state Si and the pursued final state Sf.
- Step 3. To apply a sequence of heuristic operators {HO?(Si , S2); HO?(S2, S3); …; HO?(S?, Sf)}based on a logical analysis of the operators that can transform the initial state S i in the pursued final state Sf.
- Step 4. To evaluate the level of compliance of the pursued final state S f, regarding the design objectives {DOj, …} and design restrictions {DRk, …}.

The first step was selecting the Design Components from the DTSs, with the first batch of DCs the research team talked about the importance of every single component. The third iteration removed DCs that were already covered by DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) and complemented with DCs from other DTSs.

Appendix 10.2 has all the information about this process, with the first and second iterations of the selected Design Components. Tables 27, 28, and 29 display the final selected DCs for roles, phases/activities, and artifacts. Figure 32 displays the final BPMS Methodology with all selected Desing Components.

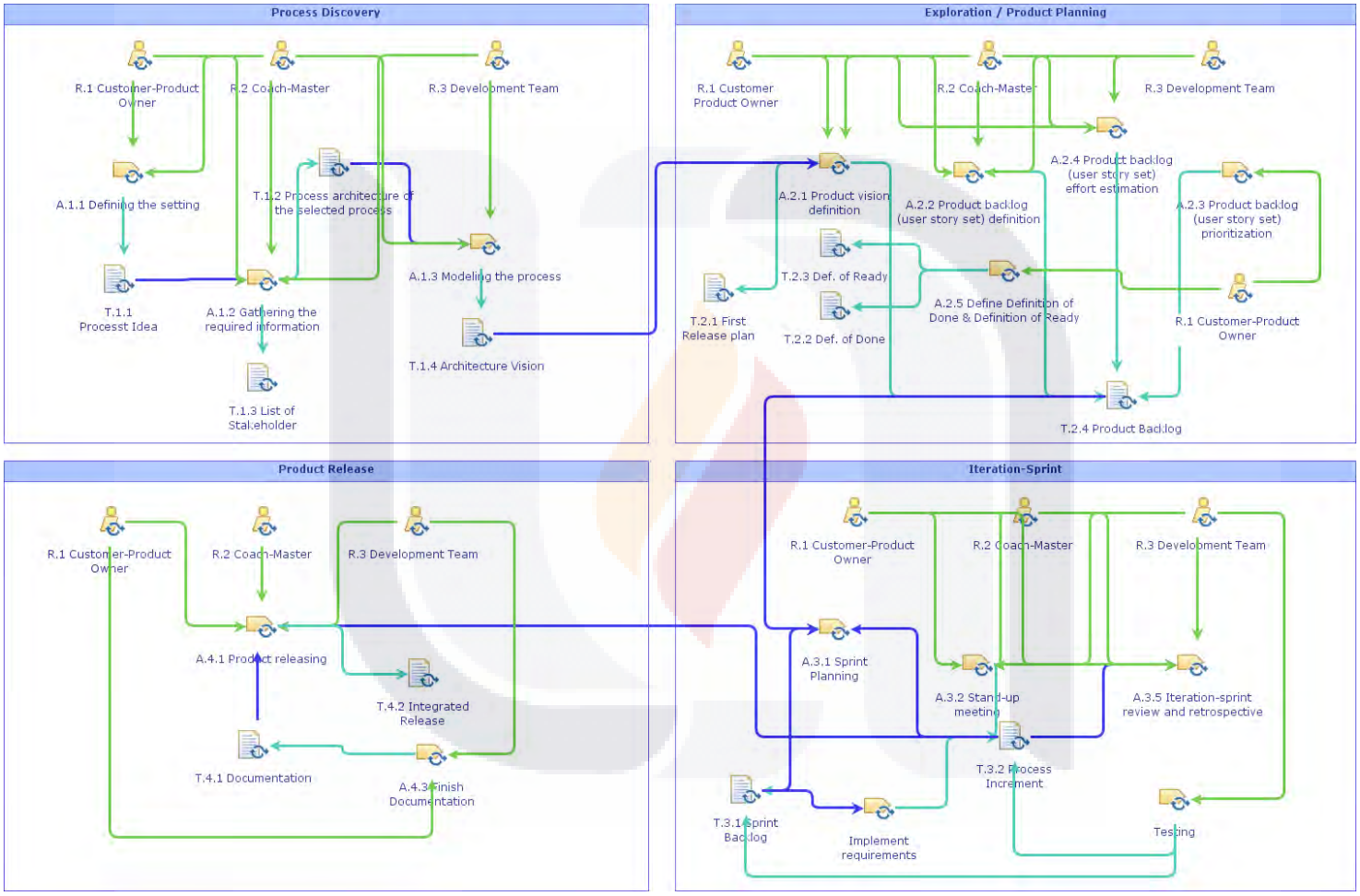


Figure 32 - BPMS Methodology Conceptual Map.

Table 27 - Final Design Components for roles.

Roles						
Design Component	**Source**	**Why this could be helpful**	**SDLC that is also using it**			
			DTS.1	**DTS.2**	**DTS.3**	**DTS.4**
DC.4 Scrum-XP Roles	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**R.1 Customer-Product Owner**: The closest role to the stakeholders, is the person who knows how to provide value to the project.	X	X	X	X
		R.2 Coach-Master: The person who is in charge of removing all the obstacles, coaching the team, ensuring transparency, and promoting self-organization.	X	X	X	X
		R.3 Development Team: The cross-functional team can build the increment every sprint. It is self-organized.	X	X	X	X

Table 28 - Final Design Components for Phases and Activities.

Design Component	**Source**	**Why this could be helpful**	**SDLC that is also using it**			
			DTS.1	**DTS.2**	**DTS.3**	**DTS.4**
DC.1 The BPM Lifecycle Phases	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	**Phase 1 - Process Discovery**: Define the team, get the information of the process, and ensure the quality.	X	X		
DC.2 The BPM Lifecycle Activities	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	**Activity A.1.1 Defining the setting**: Build the team to work on the process.	X			
DC.2 The BPM Lifecycle Activities	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	**Activity A.1.2 Gathering the required information**: Get all the needed information to work on different processes.	X	X	X	X
DC.2 The BPM Lifecycle Activities	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	**Activity A.1.3 Modeling the process**: Start to model the processes using BPMN (Business Process Management Notation).	X		X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Phase 2 - Exploration / Product Planning**: Plan all the projects and identify the project's needs.		X	X	X

DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.2.1 Product vision definition**: To Have a clear vision of the product and what needs to be developed.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.2.2 Product backlog (user story set) definition**: Create the user stories or tasks that need to be developed.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.2.3 Product backlog (user story set) prioritization**: Set the user stories to prioritize the tasks for the ones that provide more value.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.2.4 - Product backlog (user story set) effort estimation**: Estimate every single user story by the developer, it is possible to use fixed time or user story points (recommended).		X	X	X
DC.9 APBPM Activities	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Activity A.2.5 Define Definition of Done & Definition of Ready**: Create the Definition of Done and Ready. The definition of Done is all the parameters needed to accept the tasks as completed. The definition of Ready is the list of parameters that need to be met for considering a task as ready to be developed.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Phase 3 - Iteration-Sprint**: Build the increment in an Iterative process,		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.3.1 Sprint Planning**: Select the most valuable user stories to be developed during the sprint by the Product Owner. The development team chooses the task according to their skills.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.3.2 Stand-up meeting**: Meet with the team to talk about the progress, the upcoming work, and any block that can have.		X	X	X
DC.9 APBPM Activities	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Activity A.3.3 Implement requirements**: Develop every single user story.		X	X	X
DC.12 ABPM Activities	DTS.4 ABPM (Rosing and Gill, 2015)	**Activity A.3.4 Testing**: Test every single requirement that is developed during the sprint.		X	X	X

DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.3.5 Iteration-sprint review and retrospective**: Conduct a retrospective by all the team to know how what is working, and what is not. and how to be better in the next sprints.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Phase 4 - Product Release**: Release the increment with the most important features chosen by the Owner.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Activity A.4.1 Product releasing**: Release the increment.		X	X	X
DC.9 APBPM Activities	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Activity A.4.3 Finish Documentation**: Create the final documentation for the increment.		X	X	X

Table 29 - Final Design Components for Phases and Artifacts.

Design Component	Source	Why this could be helpful	SDLC that is also using it			
			DTS.1	DTS.2	DTS.3	DTS.4
DC.1 The BPM Lifecycle Phases	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	**Phase 1 - Process Discovery**: Define the team, get the information of the process, and ensure the quality.				
DC.3 The BPM Lifecycle Artifacts	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	**Artifact T.1.1 Process Idea:** A document that clearly defines the process idea**.**	X			
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.1.2 Process architecture of the selected process:** The final document of the architecture of the project.		X	X	X
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.1.3 List of Stakeholders**: A document having a list of all stakeholders of the project.		X	X	X
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.1.4 Architecture Vision**: A document with the vision of the architecture of the project.		X		
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Phase 2 - Exploration / Product Planning**: Plan all the projects and identify the project's needs.				
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.2.1 First Release Plan**: A document that details the release plan for the project.		X	X	X

DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.2.2 Def. of Done**: A list of parameters that tasks need to be met for considering tasks as done.		X	X	X
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.2.3 Def. of Ready**: A list of parameters that tasks need to be met for consideration as ready for development.		X	X	X
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.2.4 Process Backlog / Product Backlog / Story Map**: The backlog of tasks to be developed.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Phase 3 - Iteration-Sprint**: Build the increment in an Iterative process,				
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.3.1 Sprint Backlog / Story Map**: The list of tasks to be developed during the sprint.		X	X	X
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.3.2 Process Increment**: The result of merging newly developed stories with the past increment.		X	X	X
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	**Phase 4 - Product Release**: Release the increment with the most important features chosen by the Owner.				
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	**Artifact T.4.1 Documentation**: A document with the final results of sprint review, and sprint retrospective.		X	X	X
DC.13 ABPM Artifacts	DTS.4 ABPM (Rosing and Gill, 2015)	**Artifact T.4.2 Integrated Release**: The final release with the final increment.		X	X	X

5 EVALUATION OF RESULTS

5.1 EVALUATION OF AGULEBPM METHODOLOGY DOCUMENT

With the AgileBPM Methodology done and ready, it is time to validate the artifact following the DSRM step 5 from Design Science Research Methodology (DSRM) (Peffers et al., 2007). A “Validation of Experts” (Beecham, 2005) was used by different Software Engineering studies (Saadatmand, 2024); (Abdurrahman et al., 2024).

This validation technique is necessary for validating the artifact. We consider “validity of the content” as “the overall level of veracity and congruence with the overall purpose of the content” (Phillips-Wren et al., 2009). This definition suggests that 'valid content' is expected to ultimately serve its intended purpose and meet a reasonable standard of accuracy. It is akin to the concept of a model, where no entity being validated can achieve 100% accuracy. This is because any model is merely a partial representation of a real-world scenario, and it is impossible to create a model that perfectly mirrors reality.

In this section, a 'content validity' technique was employed using a Panel of Experts, following approaches commonly used in simulation (Sargent, 2013). As Sargent (Sargent, 2013, p.323) states: 'Conceptual model validation involves verifying that the theories and assumptions underlying the conceptual model are correct and that the model’s representation of the problem entity is “reasonable” for its intended purpose.'

The steps followed for this validation were the following:

1. **To have ready the textual document to be validated**. A PDF document was elaborated.

2. **To define the criteria for expert inclusion**. These criteria were defined as the people that have SENIOR expertise.

3. **To have ready a suitable questionnaire to be applied to the Panel of Experts**. This questionnaire was taken from Mora (Phillips-Wren et al., 2009). This questionnaire contains two constructs: C1 theoretical validity, and C2 theoretical consistency. The C1 contains 3 items, and the C2 contains 5 items.

4. **To define a list of potential experts to be contacted.** A list of 30 international academics and professionals in the discipline of software engineering

and BPMS people were contacted to read the AgileBPM Methodology in a given time of 3 weeks. Table 30 reports the demographic data of the sample of 15 seniors that fulfilled the required experience as asked in point 2.

Table 30 - Demographic Data of the Panel of Experts

Demographic Item	Junior Evals. Block %	Senior Evals. Block %	Junior Evals. Block %	Senior Evals. Block %
1. Age range:				
( ) <=30 years	**3**	**0**	**10%**	**0%**
( ) 31-40 years	**8**	**4**	**27%**	**13%**
( ) 41-50 years	**1**	**6**	**3%**	**20%**
( ) > 50 years	**3**	**5**	**10%**	**17%**
2. Highest academic level:				
( ) Bachelor level	**4**	**0**	**13%**	**0%**
( ) Bachelor plus Professional Certifications	**4**	**5**	**13%**	**17%**
( ) Graduate student	**2**	**7**	**7%**	**23%**
( ) Graduate completed level	**5**	**3**	**17%**	**10%**
3. Main area of formal studies:				
( ) Computer Systems / Informatics	**12**	**14**	**40%**	**47%**
( ) Business Management	**1**	**1**	**3%**	**3%**
( ) Other professional field	**2**	**0**	**7%**	**0%**
4. Main work setting:				
( ) Business enterprise	**7**	**12**	**23%**	**40%**
( ) University/Research Unit	**5**	**2**	**17%**	**7%**
( ) Government Unit	**3**	**1**	**10%**	**3%**
5. Scope of work setting:				
( ) Regional	**9**	**3**	**30%**	**10%**
( ) Nationwide	**4**	**4**	**13%**	**13%**
( ) Worldwide	**2**	**8**	**7%**	**27%**
6. Region of working setting:				
( ) USA/CAN	**2**	**7**	**7%**	**23%**
( ) Europe / Asia	**0**	**0**	**0%**	**0%**
( ) Latin America	**13**	**8**	**43%**	**27%**
7. Years in work settings:				

( ) 1-5 years	**1**	**0**	**3%**	**0%**
( ) 6-10 years	**6**	**0**	**20%**	**0%**
( ) 11-15 years	**4**	**2**	**13%**	**7%**
( ) 16-20 years	**1**	**6**	**3%**	**20%**
( ) 20 or more years .	**3**	**7**	**10%**	**23%**
8. Main Work Position:				
( ) Academic/Researcher	**6**	**2**	**20%**	**7%**
( ) IT Project Manager / IT Consultant	**8**	**7**	**27%**	**23%**
( ) Business Manager / Business Consultant	**1**	**3**	**3%**	**10%**
( ) IT Senior Developer	**0**	**3**	**0%**	**10%**
9A. Years involved (i.e. knowing, using, teaching, investigating or giving consulting) on AGILE PROCESS (SCRUM and/or XP):				
( ) <1 year	**0**	**0**	**0%**	**0%**
( ) 1-3 years	**4**	**0**	**13%**	**0%**
( ) 4-6 years	**6**	**0**	**20%**	**0%**
( ) 7-9 years	**3**	**0**	**10%**	**0%**
( ) 10 or more years	**2**	**15**	**7%**	**50%**
9B. Years involved (i.e. knowing, using, teaching, investigating or giving consulting) on BPMS (Business Process Management Systems) PRACTICES:				
( ) <=5 years	**4**	**0**	**13%**	**0%**
( ) 6-10 years	**6**	**0**	**20%**	**0%**
( ) 11-15 years	**4**	**3**	**13%**	**10%**
( ) 16-20 years	**0**	**4**	**0%**	**13%**
( ) >20 years	**1**	**8**	**3%**	**27%**
10A. Number of projects (academic, training or consulting ones) involved with AGILE PROCESS (SCRUM and/or XP):				
( ) 1-3	**3**	**0**	**10%**	**0%**
( ) 4-6	**5**	**0**	**17%**	**0%**
( ) 7-9	**5**	**0**	**17%**	**0%**
( ) 10 or more	**2**	**15**	**7%**	**50%**
10B. Number of projects (academic, training or consulting ones) involved on BPMS (Business Process Management Systems):				
( ) 1-3	**5**	**0**	**17%**	**0%**
( ) 4-6	**6**	**0**	**20%**	**0%**
( ) 7-9	**2**	**0**	**7%**	**0%**
( ) 10 or more	**2**	**15**	**7%**	**50%**

11A. Self-evaluation on the expertise level on AGILE PROCESS (SCRUM and/or XP):				
( ) very high level of expertise	**3**	**12**	**10%**	**40%**
( ) high level of expertise	**4**	**3**	**13%**	**10%**
( ) moderate level of expertise	**7**	**0**	**23%**	**0%**
( ) low level of expertise	**1**	**0**	**3%**	**0%**
( ) very low level of expertise	**0**	**0**	**0%**	**0%**
11B. Self-evaluation on the expertise level on BPMS (Business Process Management Systems):				
( ) very high level of expertise	**0**	**8**	**0%**	**27%**
( ) high level of expertise	**3**	**7**	**10%**	**23%**
( ) moderate level of expertise	**10**	**0**	**33%**	**0%**
( ) low level of expertise	**2**	**0**	**7%**	**0%**
( ) very low level of expertise	**0**	**0**	**0%**	**0%**

5. **To calculate the level of reliability, convergence validity, and discriminant validity of the 2 constructs C1 and C2 used in the applied questionnaire**. We use the PLS statistical technique (Esposito et al., 2010) due to the small data. The composite reliability index indicates the reliability and the convergent validity with factor loadings and, finally, discriminant validity using AVE (average variance extracted for each construct). Esposito et al. (2010) and Wong (2013) recommend minimal value ranges of 0.60-0.70 for reliability, 0.60-0.70 for convergent validity, and at least 0.50 for discriminant validity of the constructs. Additionally, in the test of convergent validity, each factor loading must be the greatest value in its construct regarding the other factor loading values. In the test of discriminant validity, the square root of each AVE (average variance extracted) of each construct must be greater than the correlations among constructs. It is verified in the correlation matrix where the values in the diagonal (i.e. the square roots of the AVEs) must be at least 0.70 and greater than the other values in the off-diagonal. The values obtained for each construct were satisfactory as shown in Table 31. The calculations were obtained using a free student license from the software tool SmartPLSv4 (https://www.smartpls.com).

Table 31 - Reliability and Validity of Constructs C1 and C2

		C1 THEORETICAL VALIDITY	C2 THEORETICAL CONSISTENCY
COMPOSITE RELIABILITY INDEX >= 0.60		**0.923**	**0.897**
CONVERGENT VALIDITY OF CONSTRUCT (FACTOR LOADING FOR EACH ITEM >= 0.40)	**ITEM 1**	0.796	0.884
	ITEM 2	0.749	0.95
	ITEM 3	0.534	0.894
	ITEM 4	0.867	0.727
	ITEM 5	0.908	0.586
	ITEM 6	0.926	0.772
DISCRIMINANT VALIDITY OF CONSTRUCT (SQUARE ROOT OF AVE >= 0.70)	**C1 THEORETICAL VALIDITY**	**0.901**	0.782
	C2 THEORETICAL CONSISTENCY	0.782	**0.910**

Most loadings are above 0.8, with just one item in C2 slightly lower but still within acceptable limits. The AVE values for both constructs (0.888 for C1 and 0.8 for C2) indicate that each construct captures a large portion of the variance in its respective items, affirming their reliability and validity. This confirms that both constructs are robustly measured by their items, with good convergent validity based on their high AVE values.

6. **To calculate the median, mean, and standard deviation of each item in the questionnaire**. Using a Likert scale from 1 to 5 as available options where 1 is the most negative and 5 is the most positive. Table 32 displays the obtained values.

Table 32 - Mean, Median, and Standard Deviation of the Constructs/Items C1 and C2.

Construct	Item	Mean	Median	Standard Deviation
CV1	**1**	4.667	5	0.596
CV2	**2**	4.533	5	0.718
CV3	**3**	4.667	5	0.596
CV4	**4**	4.667	5	0.596
CV5	**5**	4.533	5	0.618
CV6	**6**	4.533	5	0.618
CV7	**7**	4.8	5	0.4

Both constructs show high mean and median values close to 5, indicating overall positive responses across items. Most items have standard deviations below 0.7, suggesting reasonable consistency in responses, with some minor variations, particularly in CV2's Items 5 and 6. Items 3 (CV1) and 7 (CV2) stand out with the lowest standard deviations, indicating high consistency and reliability within their constructs. This distribution analysis suggests that both constructs are favorably rated with high consistency among items, providing a strong foundation for reliability in these constructs.

Furthermore, a one-sample, one-tailed t-test was conducted with the null hypotheses H0.1: 'The mean of Construct C1 is less than or equal to 3.0' and H0.2: 'The mean of Construct C2 is less than or equal to 3.0.' The free statistical software MaxStatLite (www.maxstatlite.com) was used for this analysis. Both null hypotheses were rejected, indicating that the means for Constructs C1 and C2 are satisfactory. Table 33 presents these results.

Table 33 - Null Hypotheses Tests on Means of Constructs C1 and C2.

Null Hypothesis	Mean of Construct	Std.Dev of Construct	P-Value	Reject H0?
H0.1 "The mean of the construct C1 is less or equal to 3.00"	4.622	0.071	0.00033	**Yes**
H0.2 "The mean of the construct C2 is less or equal to 3.00"	4.633	0.335	0.0000082	**Yes**

7. **To assess the level of validity reached by the AgileBPM Methodology document**. Based on the results of reliability and validity (convergent, and discriminant) of the instrument used to measure the theoretical validity perceived by a panel of experts, and results obtained on the means of the constructs C1 and C2, it can be assessed that the AgileBPM Methodology document is considered theoretically valid, and thus, it can be used as a source document for elaborating an AgileBPM Methodology EPG.

5.2 EMPIRICAL USABILITY EVALUATION OF AGILEBPM METHODOLOGY.

The AgileBPM Methodology was documented in an Electronic Guide using HTML and hosted in the website https://bit.ly/42s6f6L so this URL was shared with academics, and practitioners with a questionnaire taken from Gary C. et al. (1991), Karahanna et al. (1999). The constructs of interest to be evaluated for the sample of international academics and practitioners are shown in Table 34.

We got the participation of 32 practitioners, and academics from Latin America, the United States, and Canada. All the participants had more than 6 years of experience and 59% worked for a business enterprise. 59% of the participants had also a master's and PhD.

We provided the applicants with some time to read the AgileBPM Methodology documentation and check the templates. Finally, we applied two questionaries, both with the same questions, the first one related to AgileBPM Methodology, and the second one related to another BPMS Methodology that the applicant had some experience. Table 35 shows the results of the questionnaire for AgileBPM Methodology while Table 34 shows the results of the questionnaire for another BPMS methodology that applicants had experienced. The results are favorable to the AgileBPM Methodology in five constructs USEFULNESS, EASE OF USE, COMPATIBILITY, VALUE, and ATTITUDE.

Table 34 - Constructs to be Evaluated for the Sample of International Academics and Practitioners on the AgileBPM Methodology

CONSTRUCT	ITEMS	SCALE
USEFULNESS – is the degree to which using the new TOOL is perceived as being better than using the current used TOOL.	4	5-points Likert (1: strongly disagree to 5: strongly agree)
EASE OF USE - is the degree to which using the new TOOL is perceived as being free of effort.	3	5-points Likert (1: strongly disagree to 5: strongly agree)
COMPATIBILITY - is the degree to which using a new TOOL is perceived as compatible with what people do.	3	5-points Likert (1: strongly disagree to 5: strongly agree)
VALUE - the degree to which using the new TOOL is perceived as a value delivery entity for users by savings on money, time, and the provision of a variety of valuable resources, and by an overall value.	4	5-points Likert (1: very low to 5: very high)
ATTITUDE - it reflects the individual's positive and negative evaluations of performing the behavior (of adopting the evaluated artifact).	3	7-point Semantic differential scale (-3 to +3)

We got the participation of 32 practitioners, and academics from Latin America, the United States, and Canada. For this evaluation, we also applied the same

criteria that previous section. The SENIOR filtering were applied and we got 15 people for this set of data.

We provided the applicants with some time to read the AgileBPM Methodology documentation and check the templates. Finally, we applied two questionnaires, both with the same questions, the first one related to AgileBPM Methodology, and the second one related to another BPMS Methodology with the applicant had some experience.

Figure 33 and Figure 34 displays the PLS model used for calculations for the Agile Methodology and the Other Methodology known by the user.

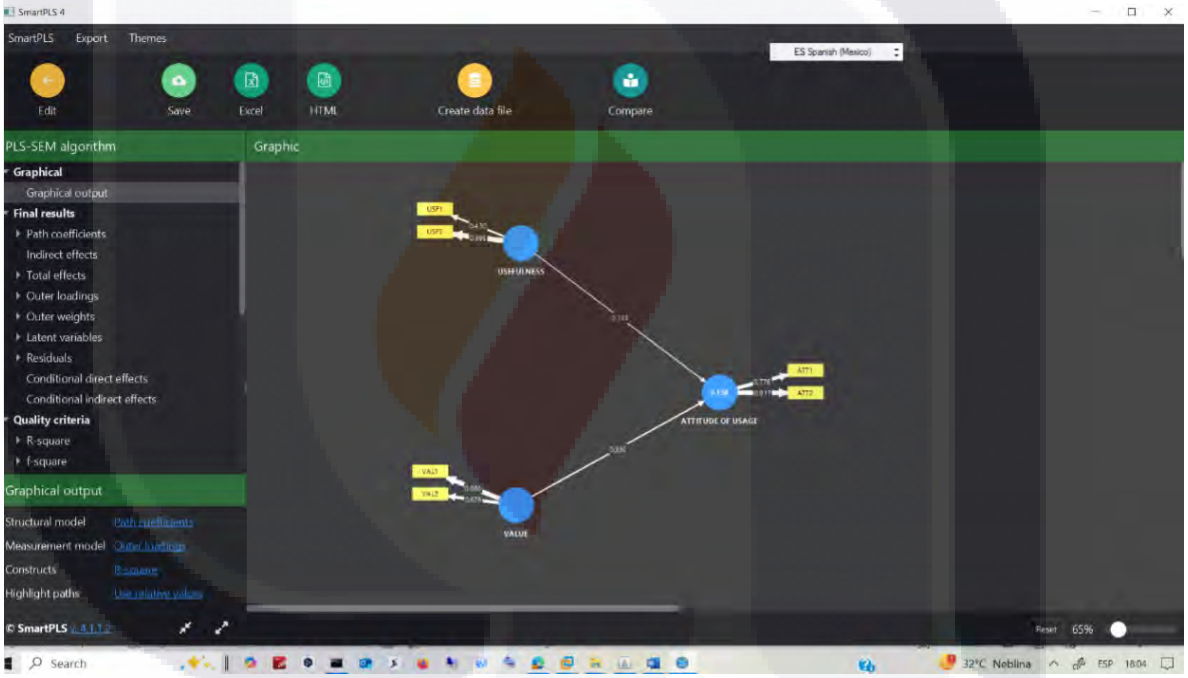


Figure 33 - PLS Model for Agile Methodology.

Figure 34 - PLS Model for other methodology.

Table 35 and Table 36 display the descriptive statistics, reliability, and discriminant validity results for the AgileBPM and the alternative methodology, respectively, based on the evaluation dataset. Descriptive statistics—median, mean, and standard deviation—were computed using the free JASP software (JASP, 2025). Reliability (Cronbach's alpha and composite reliability index) and discriminant validity (average variance extracted, AVE) were assessed using the academic version of SmartPLS v4 (SmartPLS, 2025). The results provide supporting evidence that the four final constructs—USEFULNESS, VALUE, and ATTITUDE OF POTENTIAL USAGE—were measured with acceptable reliability and discriminant validity, following established guidelines (Barclay et al., 1995; Chin, 1998; Russo & Stol, 2021). In both tables, the construct COMPATIBILITY, and EASE OF USE were excluded due to inadequate reliability and validity indicators.

Table 35 - Reliability and descriptive statistics for Agile Methodology.

Construct	Median	Mean	Std Dev.	Cronbach´s Alpha >= 0.50	Composite Reliability Index >= 0.70	Average Variance Extracted (AVE) >= 0.500
USEFULNESS	4.00	4.33	0.61	0.574	0.712	0.588
VALUE	4.00	4.27	0.52	0.633	0.815	0.695
ATTITUDE OF POTENTIAL USAGE	1.00	1.40	1.04	0.631	0.833	0.722

Table 36 - Reliability and descriptive statistics for Other Methodology.

Construct	Median	Mean	Std Dev.	Cronbach´s Alpha >= 0.50	Composite Reliability Index >= 0.70	Average Variance Extracted (AVE) >= 0.500
USEFULNESS	2.00	2.00	0.64	0.828	0.894	0.810
VALUE	2.0	2.20	0.85	0.797	0.881	0790
ATTITUDE OF POTENTIAL USAGE	-1.00	-1.33	0.84	0.542	0.733	0.607

Table 37 and Table 38 represent the complementary discriminant validity statistics for the AgileBPM and the alternative methodology, respectively, based on the evaluation dataset. These calculations were performed using the free academic version of SmartPLS v4 software (SmartPLS, 2025). The results from both tables provide supporting evidence that the four final constructs USEFULNESS, VALUE, and ATTITUDE OF POTENTIAL USAGE, demonstrate satisfactory discriminant validity (Barclay et al., 1995; Chin, 1998; Russo & Stol, 2021). In both tables, the diagonal values—representing the square root of the AVE for each construct—exceed the corresponding off-diagonal values, indicating that each construct shares more variance with its own items than with those of other constructs, as recommended by Barclay et al. (1995).

Table 37 - Discriminant Validity of the Usability Constructs for the AgileBPM

	ATTITUDE OF POTENTIAL USAGE	USEFULNESS	VALUE
ATTITUDE OF POTENTIAL USAGE	**0.850**	0.178	0.348
USEFULNESS	0.178	**0.767**	0.138
VALUE	0.348	0.138	**0.834**

Table 38 - Discriminant Validity of the Usability Constructs for the other methodology

	ATTITUDE OF POTENTIAL USAGE	USEFULNESS	VALUE
ATTITUDE OF POTENTIAL USAGE	**0.779**	0.329	0.210
USEFULNESS	0.329	**0.900**	0.299
VALUE	**0.210**	0.299	**0.889**

Table 39 and Table 40 present the convergent validity statistics for the AgileBPM and the alternative methodology, respectively, based on the evaluation dataset. These values were computed using the free academic version of SmartPLS v4 software (SmartPLS, 2025). The results provide strong evidence of adequate convergent validity for the four final constructs—USEFULNESS, VALUE, and ATTITUDE OF POTENTIAL USAGE—following established criteria (Barclay et al., 1995; Chin, 1998; Russo & Stol, 2021). As shown in both tables, the item loadings (i.e., correlations between items and their corresponding constructs) exceed 0.700 and are higher than their cross-loadings (i.e., correlations with items from other constructs), confirming satisfactory convergent validity (Barclay et al., 1995).

Additionally, four hypothesis tests were conducted to evaluate whether the AgileBPM was perceived more positively in terms of the four usability constructs compared to the alternative methodology. Given the unsatisfactory normality test results, the non-parametric Wilcoxon Matched-Pairs Signed-Rank test was applied (Sheskin, 2000). Table 41 presents these results, which were calculated using the free JASP software (JASP, 2025). The findings indicate that evaluators perceived the alternative methodology as offering better usability than the BDAS SDLC.

Table 39 - Convergent Validity of the Usability Constructs for the AgileBPM

	ATTITUDE OF USAGE	USEFULNESS	VALUE
ATT1	0.776	0.202	0.176
ATT2	0.917	0.123	0.379
USF1	0.017	0.43	0.152
USF2	0.168	0.996	0.117
VAL1	0.373	0.143	0.956
VAL2	0.131	0.066	0.676

Table 40 - Convergent Validity of the Usability Constructs for other methodology.

	ATTITUDE OF USAGE	USEFULNESS	VALUE
ATT1	0.993	-0.348	-0.209
ATT2	0.477	0	0.099
USF1	-0.087	0.8	0.024
USF2	-0.368	0.99	0.347
VAL1	-0.235	0.284	0.985
VAL2	-0.064	0.273	0.781

Table 41 - Wilcoxon Signed-Rank Tests for the Usability Constructs in AgileBPM vs alternative methodology.

Null Hypothesis	AgileBPM Median (med.1)	Alternative Methodology Median (med.2)	P-value	Implication
H0.1 For USEFULNESS construct (med.1<= med.2)	4.00	2.00	< 0.001	H0.1 is rejected, and thus the USEFULNESS of AgileBPM is better.
H0.2 For VALUE construct (med.1<= med.2)	4.00	2.00	< 0.001	H0.2 is rejected, and thus the VALUE of AgileBPM is better.
H0.3 For ATTITUDE OF POTENTIAL USAGE construct (med.1<= med.2)	1.00	-1.00	< 0.001	H0.3 is rejected, and thus the ATTITUTE OF POTENTIAL USAGE of AgileBPM is better.

5.3 APPLICATION OF THE AGUILEBPM METHODOLOGY.

To test The AgileBPM Methodology, a case demo was built using a real business process from a small business. The business process is an Expenses Claim app that has three main roles: 1) Claimer who can create new expense claims, 2) Approver the person who checks the claims and can request more information from the claimer and approve/reject the claim. 3) Finance will receive the approved claim from the Approver and will verify or reject the claim. Figure 33 shows the business process diagram.

First, the Process Discovery phase started to define all the project needs before starting the development process. In Activity - A.1.1 Defining the Setting the template F.1.01 - Process Idea was followed defining basic information from the process such as process context, process roles, and process flow. There are two templates for Activity - A.1.2 Gathering the Required Information: 1) F.1.02 - Process architecture of the selected process. 2) F.1.03 - List of Stakeholders. Both templates help with following the activity step by step with the Purpose and

Objectives, the Process Description, and Process Requirements. It is important to know that the templates provide a lot of requirements to be filled out, however not all projects need the same information, and the practitioners could avoid or add more information accordingly. In Activity - A.1.3 Modeling the Process is the activity where the process takes form and is ready to continue with the next phase. In this activity, the template F.1.04 - Architecture Vision was used to fill out the needed information like the functional/nonfunctional requirements, the description of the selected technology, the solution overview, the agreement for the sprints, and the AgileBPM roles description.

In the Exploration / Product Planning phase, there are five activities. Activity - A.2.1 Product Vision Definition - use the F.1.04 - Architecture Vision template for elaborating a release plan for the process and a backup plan. In this activity was also defined the sprint length and the timeline for the development. Activity - A.2.2 Define Definition of Done & Definition of Ready - uses templates F.2.02 -Definition of Done and F.2.03 – Definition of Ready for defining both concepts to know when a task is ready to work on and a task is completed. Activity - A.2.3 Product Backlog (user story set) Definition – the initial backlog was created. The three roles worked together to create the backlog with user stories that covered the functionality of the business process, for this activity, we used the F.2.04 – Process Backlog. Template, online tools like Jira or Trello provide better tools for replacing the template and project management. Activity - A.2.4 Product backlog (user story set) Prioritization In this activity the created backlog was prioritized from the most important to the less important. Finally, the Activity - A.2.5 - Product Backlog (user story set) Effort Estimation The development team estimated every single user story using the story points method.

During the Iteration-Sprint phase we implemented the activities Activity - A.3.1 Sprint Planning, Activity - A.3.2 Stand-up meeting, Activity - A.3.3 Implement requirements, Activity - A.3.4 Testing, and Activity - A.3.5 Iteration-sprint review and retrospective. This phase implementation uses most of the known activities in a Scrum-XP methodology. The two sprints began with the Sprint Planning meeting,

the team agreed on what user stories could be completed during the sprint, and the Sprint Backlog was created. Every single day started with the stand-up meeting where the team discussed progress and any blocks. After finishing the development of the User Story, the testing began and provided any feedback to the developers or mark the user story as completed. Finally, the sprint finished after the Sprint Retrospective meeting for getting feedback from the team and the Sprint Review shows the increment to the stakeholders.

Finally, the Product Release phase has Activity - A.4.1 Finish Documentation, and Activity - A.4.2 Product Releasing activities were F.4.01 Process Documentation template filled out with all the details for the business process environment, servers, users, and passwords.

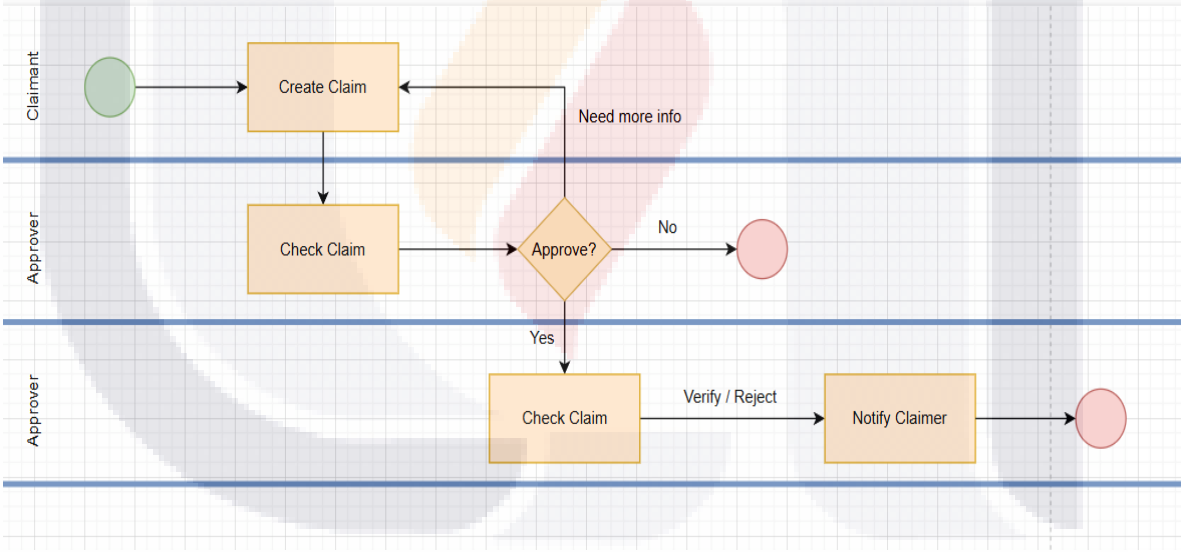


Figure 35 - BPMN diagram of the process.

The open-source BPMS/Low-Code platform Joget (https://www.jobget.com/) was used for the development of this case demo. Figure 34 displays the joget's process builder where the business process flow is created, and it is very similar to the BPMN diagram.

The advantage of using a low-code platform like Joget is that the development was very quicky without using any line of code, and most of the configuration was created on the fly using drag-and-drop tools.

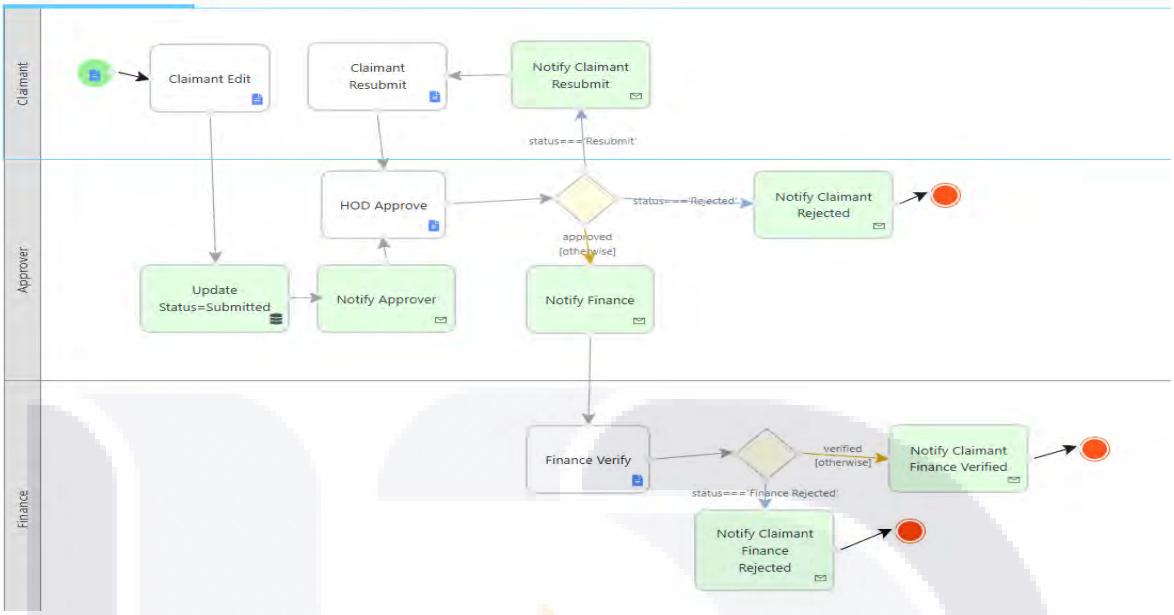


Figure 36 - Joget's process builder for the Expenses Claim app.

The advantage of using a low-code platform like Joget is that the development was very quicky without using any line of code, and most of the configuration was created on the fly using drag-and-drop tools.

6 DISCUSSION OF RESULTS

6.1 SUMMARY OF THE RESULTS

We discussed in section 1.3 the General Research Objectives (RO's), Research Questions (RQ's), and Null Hyphotesis (H0's), the Tables 42, 43, 44, and 45 provide the results got it from this research.

All the got it information for this investigations were obtained until December from 2024. These references were used for theoretically supporting and strengthing the scientific methodological validity applied to this research.

Table 42 - Results for Research Question 1

Research Question	Hypotheses	Results
RQ.1 What is the state of the art – contributions and limitations- on agile and non-agile development methodologies for Business Process Management systems?	**H0.1** There is no need for an agile development methodology for Business Process Management systems	After a Systematic Research Literature review from year 2010 to 2021 we discovered four non-agile BPMS and 4 agile BPMS methodologies. Those methodologies were studied and evaluated. After a deep analysis, we detected a lack of information and documentation about most of the methodologies. If any practitioner desires to adopt one of those methodologies, they would face several problems because there is not all the information needed to work with those methodologies. On the other hand the most important authors for Business Processes Management like Dumas (2018) refers that agility is needed in this kind of environment. For that reason, we can reject the H0.1 and sustain that new Agile Methodology for Business Process Management Systems is needed.

Table 43 - Results for Research Question 2

Research Question	Hypotheses	Results
RQ.2 What is the state of the art – capabilities, and limitations – of open-source low-code Business Process Management development platforms?	**H0.2** There are no powerful open-source low-code Business Process Management development platforms.	In section 3.1.3.2 we use Mora et al. (2016) work that compares open-source elements based on Risks Categories like Financial, Organizational, End User, and Technical. Using the tool Multi-Attribute Decision Making (MADM) for 12 open source platforms: Capgemini Open Source Maturity Model, Navica Open Source Maturity Model (OSMM), Open Business Readiness Rating (OpenBRR), Open Business Quality Rating (OpenBQR), Quality Model for Open Source Selection (QMOSS), QualOSS, Software Quality Observatory for Open Source Software model (SQO-OSS), OpenSource Maturity Model (OMM), QualiPSo—Quality Platform for Open Source Software, IRCA Model, Method for Qualification and Selection of Open Source Software (QSOSv2), and the Evaluation Framework for Free/Open Source Projects (EFFORT). After the review we found 3 viable options to work with: Joget, jBPMN, and Camunda. Using Open Decision Maker tool we found that Joget was the best open-source platform for BPMS. However there were more viable open-source platforms to work with BPMS and Low-Code in the market. For that reason, we can reject H0.2 and sustain that there are strong open-source and Low-Code platforms to work on this project.

Table 44 - Results for Research Question 3

Research Question	Hypotheses	Results
RQ.3 What elements of Agile Development and Business Process Management System Development Methodologies can be used to elaborate an Agile Business Process Management System Development Methodology that can be evaluated theoretically valid from a Panel of Experts?	**H0.3** There are no elements of Agile Development and Business Process Management System Development Methodologies that can be used to elaborate an Agile Business Process Management System Development Methodology that can be evaluated as theoretically valid by a Panel of Experts.	In section 4, we disclosed the 4 agile BPMS methodologies found in the SLR. After a deep review of the elements of each methodology, we extract all the roles, activities, and artifacts. We also added as a base the BPMS methodology from Dumas et al. (2018). Having all this information, we did an iterative process where, in the first place, we removed all the redundancies of roles, activities, and artifacts. In the second iteration, we detected the most used activities, and artifacts, as well as some key elements that could help achieve our purpose. Finally, we got al the elements from Agile BPMS methodologies that could help for construct our AgileBPM Methodology. In Section 5, we performed a questionnaire on 30 people for Latinoamerica and North America practitioners. Of those 30 persons, we did a filter by the number of worked projects and experience to get 17 people considered as experts. After conducting a statistical analysis of the questionnaire, we found that the constructs were valid and theoretically valid. For that reason, we can reject H0.3 and prove that there were elements that can help to build an Agile BPMS methodology and be evaluated as theoretically valid by a Panel of Experts.

Table 45 - Results for Research Question 4

Research Question	Hypotheses	Results
RQ.4 Can the new elaborate Agile Business Process Management System Development Methodology be documented in an Electronic Process Guide (EPG) and be evaluated as agile, useful, easy to use, compatible, and valuable by a pilot group of Software Engineering academics and practitioners?	**H0.4.1** The newly elaborated Agile Business Process Management System Development Methodology cannot be documented in an Electronic Process Guide (EPG). **H0.4.2** The newly elaborated Agile Business Process Management System Development Methodology is not considered agile, useful, easy to use, compatible, and valuable by a pilot group of Software Engineering academics and practitioners.	After creating and validating the AgileBPM Methodology, we created an electronic guide using HTML and hosted it on a website: https://bit.ly/42s6f6L. This website is public and has all the needed information for any practitioner who would like to work with the methodology. For that reason, we can reject H0.4.1. Finally, in section 5.2 we created a questionnaire for practitioners and academics where they have some time to review the new AgileBPM Methodology. After that, we apply the same questionnaire twice, once for AgileBPM and the other for any other BPMS methodology that they knew. We got positive results for the five constructs: USEFULNESS, EASE OF USE, COMPATIBILITY, VALUE, and ATTITUDE. The AgileBPM got higher Medians than other BPMS methodologies, so we can reject H0.4.2 and confirm that the AgileBPM was considered as agile, useful, easy of use, and compatible by a pilot group.

6.2 DISCUSSION ON RESULTS

AgileBPM Methodology could help practitioners from micro and small companies find a solution for developing business processes in an agile way. The Low-Code tools could also help to improve the development time. The results of this study show that a practitioner could take this AgileBPM Methodology and start working with it taking advantage of the templates that guide you during the whole process.

The Empirical Evaluation questionaries reflect that most of the participants could use the AgileBPM Methodology replacing the actual BPMS methodology that they

are implementing right now. The reason for this is that most of the Agile BPMS Methodologies lack online documentation that can help the practitioners in the process.

6.3 DISCUSSION ON CONTRIBUTIONS TO THE PRAXIS ON AGILE DEVELOPMENT FOR BPMS

AgileBPM Methodology could help practitioners from micro and small companies find a solution for developing business processes in an agile way. The Low-Code tools could also help to improve the development time. The results of this study show that a practitioner could take this AgileBPM Methodology and start working with it, taking advantage of the templates that guide you during the whole process. Agile BPM Methodology combines the planning from BPMS methodology from Dumas et al. (Dumas et al., 2018) with the best of traditional agile methodologies like Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999), 3), the best of Agile BPMS methodologies (Thiemich & Puhlmann, 2013) (Rosing & Gill, 2015).

Practitioners can start working with AgileBPM Methodology by visiting the site https://bit.ly/42s6f6L, reading the documentation, and downloading the templates, which provide a guide for what is needed during every phase and activity. As mentioned before, the templates are only a guide for the practitioner; every project could have special needs that can be included in the documentation.

6.4 LIMITATIONS

Despite the positive results of AgileBPM, limitations for future research are identified. First, its effectiveness in larger, regulated projects has not been investigated. Second, it was applied on the low-code Joget platform, and it would be useful to evaluate it on other BPMS tools, both open source and commercial. However, the results suggest that AgileBPM is a promising option for agile development.

6.5 CONCLUSIONS

The research evaluated the AgileBPM Methodology for the development of Business Process Management Systems (BPMS). Roles, activities, and artifacts were described, and a usability evaluation was conducted with international practitioners. The results indicated high scores in usability, ease of use, and

attitude toward adoption, positioning AgileBPM as a viable alternative to traditional BPMS methodologies. Key contributions include the integration of agile principles, creation of accessible documentation and empirical validation with practitioners. However, limitations were noted, such as the need for testing in larger environments. AgileBPM promises to improve efficiency and flexibility in business process management.

7 GLOSSARY

BPMS/PAIS Development Platform.

Software development platform used for designing, building, running, and monitoring a BPMS/PAIS.

Business Process Management System (BPMS) - Process-Aware Information System (PAIS)

"***Software system for supporting the operation and monitoring of a full Business Process.”*** (adapted from Reijers, 2006, p. 390)

Daily Scrum

"***The purpose of the Daily Scrum is to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work***." (Schwaber & Sutherland, 2020, p. 9)

Developers

"***Developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint***." (Schwaber & Sutherland, 2020, p. 5)

eXtreme Programming (XP)

"**XP is also a lightweight methodology or what Alistair Cockburn calls a “Crystal Methodology”. In short, methodologies of this family have high productivity and high tolerance. Communication is usually strong with short paths, especially informal (not documented). There the is only a small range**

of deliverables (artifacts), but these are delivered frequently (releases). Processes of the Crystal family identify only a few roles and activities." (Dudziak, 1999, p. 4)

Increment

"***An Increment is a concrete stepping stone toward the Product Goal. Each Increment is additive to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be usable.***" (Schwaber & Sutherland, 2020, p. 11)

Low-code

"***Application platforms that accelerate app delivery by dramatically reduce the amount of hand-coding required. Faster delivery is the primary benefit of these application platforms; they also help firms respond more quickly to customer feedback after initial software releases and provision mobile and multichannel apps. Usage of low-code platforms is gaining momentum for customer-facing applications***" (Richardson and Rymer, 2014)

Product Backlog

"***The Product Backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team***." (Schwaber & Sutherland, 2020, p. 10)

Product Owner

"***The Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. How this is done may vary widely across organizations, Scrum Teams, and individuals***." (Schwaber & Sutherland, 2020, p. 5)

Scrum

"***Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.***" (Schwaber & Sutherland, 2020, p. 3)

Scrum Master

"***The Scrum Master is accountable for establishing Scrum as defined in the Scrum Guide. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization***." (Schwaber & Sutherland, 2020, p. 6)

Scrum Team

"***The fundamental unit of Scrum is a small team of people, a Scrum Team. The Scrum Team consists of one Scrum Master, one Product Owner, and Developers. Within a Scrum Team, there are no sub-teams or hierarchies. It is a cohesive unit of professionals focused on one objective at a time, the Product Goal***." (Schwaber & Sutherland, 2020, p. 5)

Software

"*Computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system.*" (*ISO/IEC/IEEE 24765:2017(en), Systems and software engineering — Vocabulary*, 2021)

Software

"***Computer software is the product that software professionals build and then support over the long term. It encompasses programs that execute within a computer of any size and architecture, content that is presented as the computer programs execute, and descriptive information in both hard copy and virtual forms that encompass virtually any electronic media.***" (Pressman & Maxim, 2015, p. 1).

Software Engineering

"*Systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software.*" (*ISO/IEC/IEEE 24765:2017(en), Systems and software engineering — Vocabulary*, 2021)**.**

"***Encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.***" (Pressman & Maxim, 2015, p. 14).

"***Encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.***" (Pressman & Maxim, 2015, p. 14)

Software Engineering Processes

"***Software engineering processes are concerned with work activities accomplished by software engineers to develop, maintain, and operate software, such as require meets, design, construction, testing, configuration management, and other software engineering processes.***" (Abran & Moore, 2014, pp. 8–1).

Software Life Cycle

"***A software development life cycle (SDLC) includes the software processes used to specify and transform software requirements into a deliverable software product. A software product life cycle (SPLC) includes a software development life cycle plus additional software processes that provide for deployment, maintenance, support, evolution, retirement, and all other inception to retirement processes for a software product.***" (Abran & Moore, 2014, p. 8–4).

Software Process

"***A composition of phases, activities, artifacts, and resources (including the humans).***" (Oktaba & Ibargüengoitia González, 1998, p. 229)

Sprint

"***Sprints are the heartbeat of Scrum, where ideas are turned into value. They are fixed length events of one month or less to create consistency. A new Sprint starts immediately after the conclusion of the previous Sprint***." (Schwaber & Sutherland, 2020, p. 7)

Sprint Backlog

"***The Sprint Backlog is composed of the Sprint Goal (why), the set of Product Backlog items selected for the Sprint (what), as well as an actionable plan for delivering the Increment (how).***" (Schwaber & Sutherland, 2020, p. 11)

Sprint Planning

"***Sprint Planning initiates the Sprint by laying out the work to be performed for the Sprint. This resulting plan is created by the collaborative work of the entire Scrum Team***." (Schwaber & Sutherland, 2020, p. 8)

Sprint Retrospective

"***The purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness***." (Schwaber & Sutherland, 2020, p. 10)

Sprint Review

"***The purpose of the Sprint Review is to inspect the outcome of the Sprint and determine future adaptations. The Scrum Team presents the results of their work to key stakeholders and progress toward the Product Goal is discussed***." (Schwaber & Sutherland, 2020, p. 9)

Workflow Management Systems (WFMS)

"***A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications***". (van der Aalst *et al*., 2003)

8 REFERENCES

A guide to the Scrum Body of knowledge (SBOK Guide) (2013 edition). (2013). SCRUMstudy, A brand of VMEdu, Inc.

Abdurrahman, H., Nanang, S., & Choirul, H. (2024). *Sharia Retail Store Service Standards Based on Customer Preferences in the Cooperative Ecosystem | Hakim | Jurnal Aplikasi Manajemen*. https://jurnaljam.ub.ac.id/index.php/jam/article/view/8076

Abrahamsson, P., Oza, N., & Siponen, M. T. (2010). Agile Software Development Methods: A Comparative Review1. In T. Dingsøyr, T. Dybå, & N. B. Moe (Eds.), *Agile Software Development: Current Research and Future Directions* (pp. 31–59). Springer. https://doi.org/10.1007/978-3-642-12575-1_3

Abran, A., & Moore, J. W. (2014). *Guide to the software engineering body of knowledge*.

Badakhshan, P., Conboy, K., Grisold, T., & vom Brocke, J. (2019). Agile business process management: A systematic literature review and an integrated framework. *Business Process Management Journal*, *26*(6), 1505–1523. https://doi.org/10.1108/BPMJ-12-2018-0347

Barabino, G., Grechi, D., Tigano, D., Corona, E., & Concas, G. (2014). Agile Methodologies in Web Programming: A Survey. In G. Cantone & M. Marchesi (Eds.), *Agile Processes in Software Engineering and Extreme*

Programming (pp. 234–241). Springer International Publishing. https://doi.org/10.1007/978-3-319-06862-6_16

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Robert C., M., Mellor, S., Thomas, D., James, G., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Schwaber, K., & Sutherland, J. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Beecham. (2005). *Using an expert panel to validate a requirements process improvement model—ScienceDirect*. https://www.sciencedirect.com/science/article/abs/pii/S0164121204000974

Bider, I., & Jalali, A. (2016). Agile business process development: Why, how and when—applying Nonaka's theory of knowledge transformation to business process development. *Information Systems and E-Business Management*, *14*(4), 693–731. https://doi.org/10.1007/s10257-014-0256-1

Boehm, B., & Turner, R. (2003). Using risk to balance agile and plan-driven methods. *Computer*, *36*(6), 57–66. Computer. https://doi.org/10.1109/MC.2003.1204376

C., G., Moore, & Benbasat, I. (1991). *Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation | Information Systems Research*. https://pubsonline.informs.org/doi/abs/10.1287/isre.2.3.192

CMMI for Development, Version 1.3. (n.d.). Retrieved March 23, 2021, from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9661

Conboy, K. (2009). Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research*, *20*(3), 329–354. https://doi.org/10.1287/isre.1090.0236

Damij, N., Damij, T., Grad, J., & Jelenc, F. (2008). A methodology for business process improvement and IS development. *Information & Software Technology*, *50*, 1127–1141. https://doi.org/10.1016/j.infsof.2007.11.004

Denning, P. J. (1999). COMPUTER SCIENCE: THE DISCIPLINE. *Encyclopedia of Computer Science*, 9–23.

Dudziak, T. (1999). *eXtreme Programming An Overview*. Methoden und Werkzeuge der Software: produktion WS. http://csis.pace.edu/~marchese/CS616/Agile/XP/XP_Overview.pdf

Dumas, M., Aalst, W. M. van der, & Hofstede, A. H. ter. (2005). *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. John Wiley & Sons.

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). Introduction to Business Process Management. In M. Dumas, M. La Rosa, J. Mendling, & H. A. Reijers (Eds.), *Fundamentals of Business Process Management* (pp. 1–33). Springer. https://doi.org/10.1007/978-3-662-56509-4_1

Esposito Vinzi, V., Chin, W. W., Henseler, J., & Wang, H. (Eds.). (2010). *Handbook of Partial Least Squares: Concepts, Methods and Applications*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-32827-8

Galvan, S., Mora, M., & Laporte, C. Y. (2021). *Reconciliation of scrum and the project management process of the ISO/IEC 29110 standard-Entry profile—*

An experimental evaluation through usability measures | SpringerLink. https://link.springer.com/article/10.1007/s11219-021-09552-3

Garousi, V., Borg, M., & Oivo, M. (2020). Practical relevance of software engineering research: Synthesizing the community's voice. *Empirical Software Engineering*, *25*(3), 1687–1754. https://doi.org/10.1007/s10664-020-09803-0

Grand View Research. (2020). *Low-Code Application Development Platform Market Report, 2020-2027*. https://www.grandviewresearch.com/industry-analysis/low-code-application-development-platform-market

Greeno, G., J., Simon, & A., H. (1987). *Problem Solving and Reasoning*. https://apps.dtic.mil/sti/citations/tr/ADA219146

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625

Hevner, A. R., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 75–105.

Hoda, R., Salleh, N., & Grundy, J. (2018). The Rise and Evolution of Agile Software Development. *IEEE Software*, *35*(5), Article 5. https://doi.org/10.1109/MS.2018.290111318

Hong, W., Thong, J. Y. L., Chasalow, L. C., & Dhillon, G. (2011). User Acceptance of Agile Information Systems: A Model and Empirical Test. *Journal of Management Information Systems*, *28*(1), 235–272. https://doi.org/10.2753/MIS0742-1222280108

Hughes, D. L., Rana, N. P., & Simintiras, A. C. (2017). The changing landscape of IS project failure: An examination of the key factors. *Journal of Enterprise Information Management*, *30*(1), 142–165. https://doi.org/10.1108/JEIM-01-2016-0029

Humphrey, W. S. (1988). The software engineering process: Definition and scope. *Proceedings of the 4th International Software Process Workshop on Representing and Enacting the Software Process*, 82–83. https://doi.org/10.1145/75110.75122

ISO/IEC/IEEE 24765:2017(en), Systems and software engineering—Vocabulary. (2021). https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:24765:ed-2:v1:en

Javanmard, M., & Alian, M. (2015). Comparison between Agile and Traditional software development methodologies. *Cumhuriyet Üniversitesi Fen Edebiyat Fakültesi Fen Bilimleri Dergisi*, *36*(3), Article 3.

Jung, J., Choi, I., & Song, M. (2007). An integration architecture for knowledge management systems and business process management systems. *Computers in Industry*, *58*(1), 21–34. https://doi.org/10.1016/j.compind.2006.03.001

Karagiannis, D. (1995). BPMS: Business process management systems. *ACM SIGOIS Bulletin*, *16*(1), 10–13. https://doi.org/10.1145/209891.209894

Karahanna, E., Straub, D. W., & Chervany, N. L. (1999). Information Technology Adoption Across Time: A Cross-Sectional Comparison of Pre-Adoption and Post-Adoption Beliefs. *MIS Quarterly*, *23*(2), 183–213. https://doi.org/10.2307/249751

Krafzig, D., Banke, K., & Slama, D. (2005). *Enterprise SOA: Service-oriented Architecture Best Practices*. Prentice Hall Professional.

Laanti, M., Similä, J., & Abrahamsson, P. (2013). Definitions of Agile Software Development and Agility. In F. McCaffery, R. V. O'Connor, & R. Messnarz (Eds.), *Systems, Software and Services Process Improvement* (pp. 247–258). Springer. https://doi.org/10.1007/978-3-642-39179-8_22

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021, July 15). *Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective*. https://doi.org/10.1145/3475716.3475782

Markets and Markets. (2020). *Low-Code Development Platform Market Size, Share and Global Market Forecast to 2025 | MarketsandMarkets*. https://www.marketsandmarkets.com/Market-Reports/low-code-development-platforms-market-103455110.html

Martins, P. V., & Zacarias, M. (2017). An Agile Business Process Improvement Methodology. *Procedia Computer Science*, *121*, 129–136. https://doi.org/10.1016/j.procs.2017.11.018

Meziani, R., & Magalhães, R. (2009). *Proposals for an Agile Business Process Management Methodology*. 15.

Mora, M. (2009). *Metodo-Conceptual-Dr-Mora-v-2009-OK.pdf*.

Mora, M., Gómez, J. M., O'Connor, R. V., & Gelman, O. (2016). An MADM risk-based evaluation-selection model of free-libre open source software tools. *International Journal of Technology, Policy and Management*, *16*(4), 326–354. https://doi.org/10.1504/IJTPM.2016.081665

Mu, W., Bénaben, F., & Pingaud, H. (2015). A methodology proposal for collaborative business process elaboration using a model-driven approach. *Enterprise Information Systems*, *9*(4), 349–383. https://doi.org/10.1080/17517575.2013.771410

Mutschler, B., Reichert, M., & Bumiller, J. (2008). Unleashing the Effectiveness of Process-Oriented Information Systems: Problem Analysis, Critical Success Factors, and Implications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(3), 280–291. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). https://doi.org/10.1109/TSMCC.2008.919197

Nascimento, A. R. D., Baldam, R. de L., Costa, L., & Coelho Junior, T. de P. (2019). Applications of business governance and the Unified BPM Cycle in public credit recovery activities. *Business Process Management Journal*, *26*(1), 312–330. https://doi.org/10.1108/BPMJ-11-2017-0317

Navarro, A. (2009). *A SWEBOK-based Viewpoint of the Web Engineering Discipline*. 32.

Newell, A., & Simon, H. A. (1972). *Human problem solving* (Vol. 104). Prentice-hall Englewood Cliffs, NJ. http://www.sci.brooklyn.cuny.edu/~kopec/cis718/fall_2005/2/Rafique_2_humanthinking.doc

Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation. *Organization Science*, *5*(1), 14–37. https://doi.org/10.1287/orsc.5.1.14

Oktaba, H., & Ibargüengoitia González, G. (1998). *Software Process Modeled with Objects: Static View*. http://www.repositoriodigital.ipn.mx//handle/123456789/15087

Papazoglou, M. P., & van den Heuvel, W.-J. (2007). Business process development life cycle methodology. *Communications of the ACM*, *50*(10), 79–85. https://doi.org/10.1145/1290958.1290966

Parnas, D. L. (2010). Risks of undisciplined development. *Communications of the ACM*, *53*(10), 25–27. https://doi.org/10.1145/1831407.1831419

Paulk, M. C. (2002). *Agile Methodologies and Process Discipline*. https://doi.org/10.1184/R1/6620972.v1

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007a). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007b). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Petersen, K., & Wohlin, C. (2009a). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, *82*(9), 1479–1490. https://doi.org/10.1016/j.jss.2009.03.036

Petersen, K., & Wohlin, C. (2009b). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, *82*(9), 1479–1490. https://doi.org/10.1016/j.jss.2009.03.036

Phillips-Wren, G., Mora, M., Forgionne, G. A., & Gupta, J. N. D. (2009). An integrative evaluation framework for intelligent decision support systems. *European Journal of Operational Research*, *195*(3), 642–652. https://doi.org/10.1016/j.ejor.2007.11.001

Pressman, R. S., & Maxim, B. R. (2015). *Software engineering: A practitioner's approach* (Eighth edition). McGraw-Hill Education.

Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology*, *50*(4), 280–295. https://doi.org/10.1016/j.infsof.2007.02.002

Ravesteyn, P., & Batenburg, R. (2010a). Surveying the critical success factors of BPM-systems implementation. *Business Process Management Journal*, *16*(3), 492–507. https://doi.org/10.1108/14637151011049467

Ravesteyn, P., & Batenburg, R. (2010b). Surveying the critical success factors of BPM-systems implementation. *Business Process Management Journal*, *16*(3), 492–507. https://doi.org/10.1108/14637151011049467

Reijers, H. A. (2006). Implementing BPM systems: The role of process orientation. *Business Process Management Journal*, *12*(4), 389–409. https://doi.org/10.1108/14637150610678041

Richardson, C., & Rymer, J. R. (2014, June 9). *New Development Platforms Emerge For Customer-Facing Applications*. Forrester: Cambridge. https://www.forrester.com/report/New+Development+Platforms+Emerge+For+CustomerFacing+Applications/RES113411

Rodríguez, L., Mora, M., Vargas Martin, M., O'Connor, R., & Rodriguez, F. (2009). Process Models of SDLCs: Comparison and Evolution. In *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications* (pp. 76–89). https://doi.org/10.4018/978-1-59904-887-1.ch005

Rosing, M. von, & Gill, A. (2015). *Applying Agile Principles to BPM* (Vol. 1, pp. 553–577). https://doi.org/10.1016/B978-0-12-799959-3.00027-6

Rymer, J. R. (2017). *The Forrester Wave™: Low-Code Development Platforms For AD&D Pros, Q4 2017*. 21.

Saadatmand, M. (2024). *A Hierarchical Decision Model for Evaluating the Strategy Readiness of Quantitative Machine Learning/Data Science-Driven Investment Strategies—ProQuest*. https://www.proquest.com/openview/12b3151aa6a62d144519cc080f7d3bc9/1?pq-origsite=gscholar&cbl=18750&diss=y

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

Sargent, R. G. (2013). An introduction to verification and validation of simulation models. *2013 Winter Simulations Conference (WSC)*, 321–327. https://doi.org/10.1109/WSC.2013.6721430

Schwaber, K. (1997). SCRUM Development Process. In J. Sutherland, C. Casanave, J. Miller, P. Patel, & G. Hollowell (Eds.), *Business Object Design and Implementation* (pp. 117–134). Springer London. https://doi.org/10.1007/978-1-4471-0947-1_11

Schwaber, K., & Sutherland, J. (2020, January 11). *Scrum Guide | Scrum Guides*. https://scrumguides.org/scrum-guide.html

Shankarmani, R., Pawar, R., S. Mantha, S., & Babu, V. (2012). Agile Methodology Adoption: Benefits and Constraints. *International Journal of Computer Applications*, *58*(15), 31–37. https://doi.org/10.5120/9361-3698

Shaw, M. (2003). Writing good software engineering research papers. *25th International Conference on Software Engineering, 2003. Proceedings.*, 726–736. https://doi.org/10.1109/ICSE.2003.1201262

Silva, A. R., Meziani, R., Magalhães, R., Martinho, D., Aguiar, A., & Flores, N. (2009). AGILIPO: Embedding Social Software Features into Business Process Tools. In S. Rinderle-Ma, S. Sadiq, & F. Leymann (Eds.), *Business Process Management Workshops* (pp. 219–230). Springer. https://doi.org/10.1007/978-3-642-12186-9_21

State of Agile Survey. (2021, March 2). https://stateofagile.com/

Taudes, A., Feurstein, M., & Mild, A. (2000). Options Analysis of Software Platform Decisions: A Case Study. *MIS Quarterly*, *24*(2), 227–243. https://doi.org/10.2307/3250937

Thiemich, C., & Puhlmann, F. (2013). An Agile BPM Project Methodology. In F. Daniel, J. Wang, & B. Weber (Eds.), *Business Process Management* (pp. 291–306). Springer. https://doi.org/10.1007/978-3-642-40176-3_25

Turner, R. (2003). *Management Basics People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods*.

van der Aalst, W. M. P., ter Hofstede, A. H. M., & Weske, M. (2003). Business process management: A survey. *Proceedings of the 1st International Conference on Business Process Management, Volume 2678 of LNCS*, 1–12.

Vincent, P., Iijima, K., Driver, M., Wong, J., & Natis, Y. (2019). *Magic quadrant for enterprise low-code application platforms*. https://smallake.kr/wp-content/uploads/2020/01/gartner-magic-quadrant-for-enterprise-low-code-application-platforms-august-20191.pdf

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, *52*(10), 376–381. https://doi.org/10.1016/j.ifacol.2019.10.060

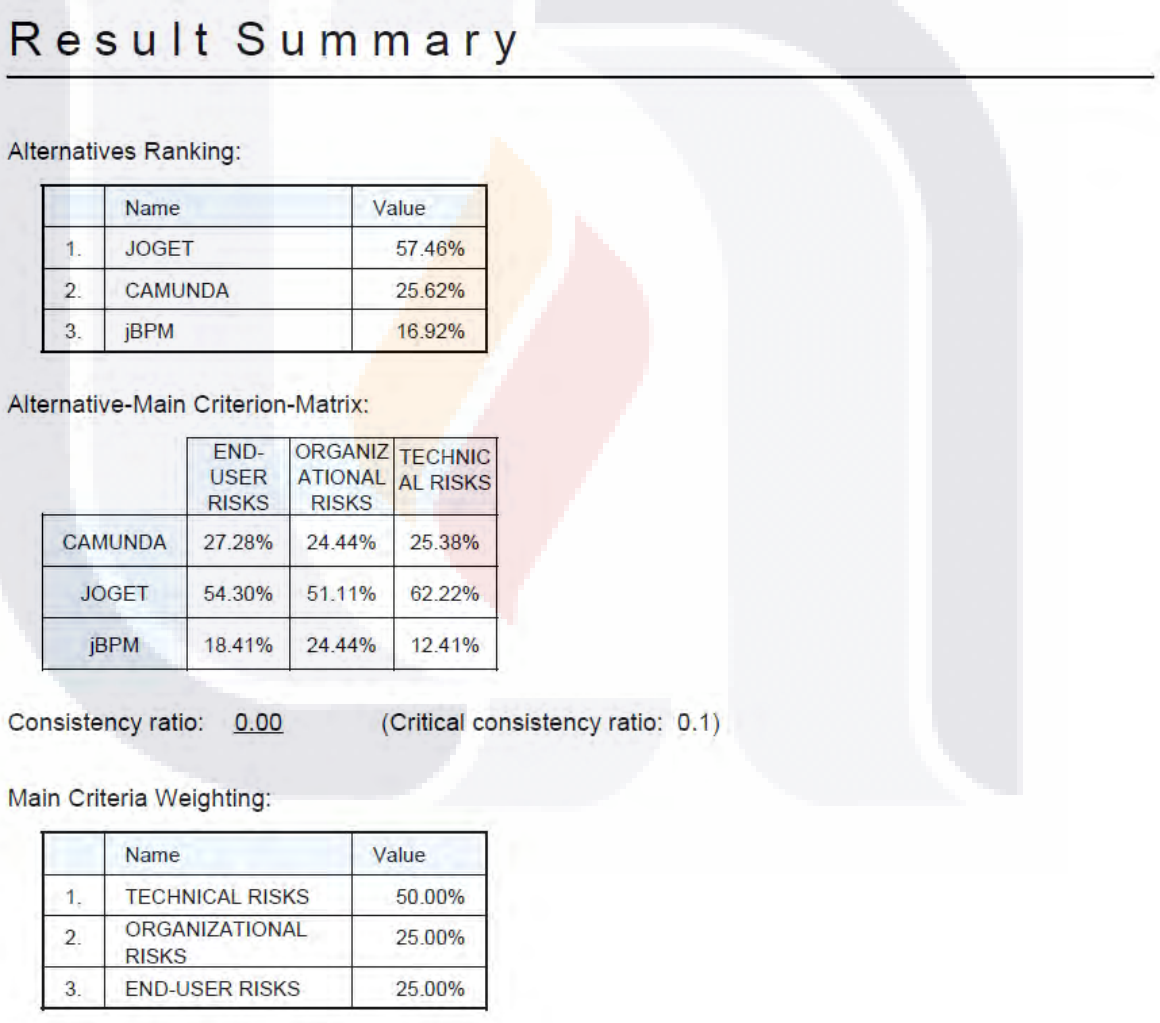
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer Science & Business Media.

Wong, K. K.-K. (2013). *Partial Least Squares Structural Equation Modeling (PLS-SEM) Techniques Using SmartPLS*.

9 APPENDIX

9.1 LOW-CODE DEVELOPMENT PLATFORM OPEN-SOURCE DECISION-MAKING RESULTS

Figures from 9–1 to 9-14 display the results from the comparison done with Open Decision Maker software on Low-Code open-source platforms JOGET, CAMUNDA, and jBPM.



Result Summary

Alternatives Ranking:

	Name	Value
1.	JOGET	57.46%
2.	CAMUNDA	25.62%
3.	jBPM	16.92%

Alternative-Main Criterion-Matrix:

	END-USER RISKS	ORGANIZATIONAL RISKS	TECHNICAL RISKS
CAMUNDA	27.28%	24.44%	25.38%
JOGET	54.30%	51.11%	62.22%
jBPM	18.41%	24.44%	12.41%

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Main Criteria Weighting:

	Name	Value
1.	TECHNICAL RISKS	50.00%
2.	ORGANIZATIONAL RISKS	25.00%
3.	END-USER RISKS	25.00%

Figure 37 - Results Summary

Criteria Summary

1. Main Criterion: ORGANIZATIONAL RISKS

Parent(s): -

Description:

Weighting Matrix:

	INTERNAL EXPERTISE	TOP MANAGEMENT SUPPORT	TRAINING
INTERNAL EXPERTISE	1	1.00	0.25
TOP MANAGEMENT SUPPORT	1.00	1	0.25
TRAINING	4.00	4.00	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	51.11%
2.	jBPM	24.44%
3.	CAMUNDA	24.44%

Figure 38 - Organizational Risks

1.1. Sub Criterion: TRAINING

Parent(s): ORGANIZATIONAL RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	0.33	1.00
JOGET	3.00	1	3.00
jBPM	1.00	0.33	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	60.00%
2.	jBPM	20.00%
3.	CAMUNDA	20.00%

Figure 39 - Training results.

1.2. Sub Criterion: TOP MANAGEMENT SUPPORT

Parent(s): ORGANIZATIONAL RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	1.00	1.00
JOGET	1.00	1	1.00
jBPM	1.00	1.00	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	33.33%
2.	jBPM	33.33%
3.	CAMUNDA	33.33%

Figure 40 - Top Management Support results.

1.3. Sub Criterion: INTERNAL EXPERTISE

Parent(s): ORGANIZATIONAL RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	1.00	1.00
JOGET	1.00	1	1.00
jBPM	1.00	1.00	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	33.33%
2.	jBPM	33.33%
3.	CAMUNDA	33.33%

Figure 41 - Internal Expertise results.

2. Main Criterion: END-USER RISKS

Parent(s): -

Description:

Weighting Matrix:

	FUNCTIONALITY-QUALITY	USABILITY	USEFULNESS-RELEVANCE
FUNCTIONALITY-QUALITY	1	0.25	1.00
USABILITY	4.00	1	4.00
USEFULNESS-RELEVANCE	1.00	0.25	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	54.30%
2.	CAMUNDA	27.28%
3.	jBPM	18.41%

Figure 42 - End-User Risks results.

2.1. Sub Criterion: FUNCTIONALITY-QUALITY

Parent(s): END-USER RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	3.00	5.00
JOGET	0.33	1	3.00
jBPM	0.20	0.33	1

Consistency ratio: 0.03 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	CAMUNDA	63.70%
2.	JOGET	25.83%
3.	jBPM	10.47%

Figure 43 - Functionality-Quality results.

2.2. Sub Criterion: USEFULNESS-RELEVANCE

Parent(s): END-USER RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	1.00	1.00
JOGET	1.00	1	1.00
jBPM	1.00	1.00	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	33.33%
2.	jBPM	33.33%
3.	CAMUNDA	33.33%

Figure 44 - Usefulness-Relevance results.

2.3. Sub Criterion: USABILITY

Parent(s): END-USER RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	0.25	1.00
JOGET	4.00	1	4.00
jBPM	1.00	0.25	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	66.67%
2.	jBPM	16.67%
3.	CAMUNDA	16.67%

Figure 45 - Usability results.

3. Main Criterion: TECHNICAL RISKS

Parent(s): -

Description:

Weighting Matrix:

	COMMUNITY SUPPORT	DOCUMENTATION	MATURITY-LONGEVITY	SECURITY-RELIABILITY
COMMUNITY SUPPORT	1	0.33	1.00	1.00
DOCUMENTATION	3.00	1	3.00	3.00
MATURITY-LONGEVITY	1.00	0.33	1	1.00
SECURITY-RELIABILITY	1.00	0.33	1.00	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	62.22%
2.	CAMUNDA	25.38%
3.	jBPM	12.41%

Figure 46 - Technical Risks results.

3.1. Sub Criterion: COMMUNITY SUPPORT

Parent(s): TECHNICAL RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	0.20	0.33
JOGET	5.00	1	3.00
jBPM	3.00	0.33	1

Consistency ratio: 0.03 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	63.70%
2.	jBPM	25.83%
3.	CAMUNDA	10.47%

Figure 47 - Community Support results.

3.2. Sub Criterion: DOCUMENTATION

Parent(s): TECHNICAL RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	0.33	3.00
JOGET	3.00	1	5.00
jBPM	0.33	0.20	1

Consistency ratio: 0.03 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	63.70%
2.	CAMUNDA	25.83%
3.	jBPM	10.47%

Figure 48 - Documentation results.

3.3. Sub Criterion: MATURITY-LONGEVITY

Parent(s): TECHNICAL RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	0.20	3.00
JOGET	5.00	1	7.00
jBPM	0.33	0.14	1

Consistency ratio: 0.06 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	73.06%
2.	CAMUNDA	18.84%
3.	jBPM	8.10%

Figure 49 - Maturity-Longevity results.

3.4. Sub Criterion: SECURITY-RELIABILITY

Parent(s): TECHNICAL RISKS

Description:

Weighting Matrix:

	CAMUNDA	JOGET	jBPM
CAMUNDA	1	1.00	5.00
JOGET	1.00	1	5.00
jBPM	0.20	0.20	1

Consistency ratio: 0.00 (Critical consistency ratio: 0.1)

Result (Ranking):

	Name	Value
1.	JOGET	45.45%
2.	CAMUNDA	45.45%
3.	jBPM	9.09%

Figure 50 - Security-Reliability results.

9.2 DESIGN OF THE ARTIFACT METHODOLOGY.

Once the Design Theoretical Sources were selected the Design Components were chosen from the Roles, Activities, and Artifacts that could help to the design of the BPMS Methodology.

The Table 46, Table 47, and Table 48 show the selected Design Components for the first selected Design Components. Once the Desing Components were selected the second iteration of the process reviewed every single component and asked about its importance to be in the BPMS Methodology.

The second iteration provides the second wave of Desing Components that could be important to be in the BPMS Methodology. The third and the last iterations were processed to have the minimal Design Components for the Methodology. It is also important to say that there were a lot of Desing Components that are using the same activities and artifacts that use the DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999) so the Desing Components for this DTS were selected and complemented with other Activities and Artifacts.

Table 46 - Roles for Desing Components first and second iterations.

Roles							
Design Component	**Source**	**Name**	**Why this could be helpful**	**SDLC that is also using it**			
				DTS.1	**DTS.2**	**DTS.3**	**DTS.4**
DC.4 Scrum-XP Roles	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	{Customer-Product Owner; Coach-Master; Development Team}	**Customer-Product Owner**: The closest role to the stakeholders, is the person who knows how to provide value to the project.	X	X	X	X
			Coach-Master: The person who is in charge of removing all the obstacles, coaching the team, ensuring transparency, and promoting self-organization.	X	X	X	X
			Development Team: The cross-functional team that can build the increment every sprint. It is self-organized.	X	X	X	X

Table 47 - Phases and Activities for Desing Components first and second iteration.

Design Component	**Source**	**Name**	**Why this could be helpful**	**SDLC that is also using it**				**Iteration**		
				DTS.1	**DTS.2**	**DTS.3**	**DTS.4**	**1**	**2**	**2**
DC.1 The BPM Lifecycle Phases	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	{Process Identification, Process Discovery}	**Process Identification:** Take all the business processes and set clear criteria for selecting specific processes for doing improvements.	X	X	X	X	X		
			Process Discovery: Define the team, get the information of the process, and ensure the quality.	X	X			X	X	X
DC.2 The BPM Lifecycle Activities	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	{Process Identification [Process architecture definition, Process selection], Process	**Process Identification - Process architecture definition**: Represents the processes that exist in an organization.	X				X		

		Discovery [Defining the setting, Gathering the required information, Modeling the process, Assuring model quality]}	**Process Identification - Process selection**: Observe the business processes to define the basis for process selection.	X				X		
			Process Discovery - Defining the setting: Build the team to work on the process.	X				X	X	X
			Process Discovery - Gathering the required information: Get all the needed information to work on different processes.	X	X	X	X	X	X	X
			Process Discovery - Modeling the process: Start to model the processes using BPMN (Business Process Management Notation).	X		X	X	X		X
			Process Discovery - Assuring model quality: Ensure that the processes modeled have the needed quality.	X				X		
DC.5 Scrum-XP Phases	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	{Exploration, Product Planning, Iteration-Sprint Planning, Iteration-Sprint, Product Release}	**Exploration**: Plan all the projects and identify the project's needs.		X	X	X	X	X	X
			Product Planning: Plan the product according to the needs.		X	X	X	X	X	X
			Iteration-Sprint Planning: Select the activities that provide more value to the project as a priority to be developed during a fixed time.		X	X	X	X	X	X
			Iteration-Sprint: Build the increment in an Iterative process,		X	X	X	X	X	X
			Product Release: Release the increment with the most important features chosen by the Owner.		X	X	X	X	X	X
DC.6 Scrum-XP Activities	DTS.2 Scrum-XP (Schwaber & Sutherland, 2020) (Dudziak, 1999)	{Exploration [product vision definition; product backlog (user story set) definition;	**Exploration - Product vision definition**: To Have a clear vision of the product and what needs to be developed.		X	X	X	X	X	X

		product backlog (user story set) prioritization; optional: spike testing]} {Product Planning [product backlog (user story set) effort estimation; product backlog (user story set) negotiation; optional: style codifying standard definition]} {Iteration-Sprint Planning [iteration-sprint user story selection; iteration sprint user story task planning iteration-sprint user story plan negotiation]} {Iteration-Sprint [stand-up meeting; customer functional tests elaboration; simple design; codification and unit testing; increment integration and customer functional testing; iteration-sprint review and retrospective]} {Product Release [product releasing]}	**Exploration - Product backlog (user story set) definition**: Create the user stories or tasks that need to be developed.		X	X	X	X	X	X
			Exploration - Product backlog (user story set) prioritization: Set the user stories to prioritize the tasks for the ones that provide more value.		X	X	X	X	X	X
			Exploration - Spike testing: Define the spikes that need some effort to have better knowledge to close the spike and create the needed user stories.		X			X		
			Product Planning - Product backlog (user story set) effort estimation: Estimate every single user story by the developer, it is possible to use fixed time or user story points (recommended).		X	X	X	X	X	X
			Product Planning - Product backlog (user story set) negotiation: Negotiate as needed in some user stories. Negotiations with the product owner can avoid conflicts during the sprint.		X		X	X	X	
			Product Planning - Style codifying standard definition: Defining standards in the code could help to create a better product and be more maintainable in the feature.		X	X	X	X		
			Iteration-Sprint Planning - Iteration-sprint user story selection: Select the most valuable user stories to be developed during the sprint by the Product Owner. The development team chooses the task according to their skills.		X	X	X	X	X	

			Iteration-Sprint Planning - Iteration-sprint user story task planning: Planning the user story selected in terms of what would be the best approach for doing this task.		X	X	X	X	X	
			Iteration-Sprint Planning - Iteration-sprint user story plan negotiation: Negotiate with the product owner some items for the Sprint Planning		X	X	X	X	X	
			Iteration-Sprint - Stand-up meeting: Meet with the team to talk about the progress, the upcoming work, and any block that can have.		X	X	X	X	X	X
			Iteration-Sprint - Customer functional tests elaboration: Elaborate test cases for every single user story that is developed.		X	X	X	X		
			Iteration-Sprint - Simple design: Create a simple design of how to develop the story.		X	X	X	X		
			Iteration-Sprint - Codification and unit testing: Code and test the selected user story.		X	X	X	X	X	
			Iteration-Sprint - Increment integration and customer functional testing: Merge the finished user's stories with increment which is a working version of the product with the functionality described in the developed user stories.		X			X	X	
			Iteration-Sprint - Iteration-sprint review and retrospective: Conduct a retrospective by all the team to know how what is working, and what is not. and how to be better in the next sprints.		X	X	X	X	X	X

			Product Release - Product releasing: Release the increment.		X	X	X	X	X	X
DC.8 APBPM Phases	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	{Project Scoping, Project Kick-Off, Sprint 0, Sprint 1-n, Release Sprint}	**Project Scoping**: Define the scope of the project.		X	X	X	X	X	
			Project Kick-Off: Define the team, create the initial release plan, and define the sprint length.		X	X	X	X	X	
			Sprint 0: Define some parameters that are going to be used in the next sprints.		X	X		X	X	
			Sprint 1-n: Run a regular Scrum sprint.		X	X	X	X	X	X
			Release Sprint: Release the documentation, the training, and the product in this phase.		X	X	X	X	X	X
DC.9 APBPM Activities	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	{Project Scoping [Define target parameters, Create project idea, Define project start/end, Identify Stakeholder, Evaluate BPM Maturity], Project Kick-Off [Define sprint length, Create initial release plan, Establish architecture vision, Build team], Sprint 0 [Define Definition of Done & Definition of Ready, Identify initial requirements, Define initial architecture, Setup project environment], Sprint 1-n [Refine process backlog, Plan sprint,	**Project Scoping - Define target parameters**: Define the most important parameters to be used during the project.		X	X	X	X		
			Project Scoping - Create project idea: Create the main idea for the project.		X	X	X	X	X	
			Project Scoping - Define project start/end: Define when the project is going to start and end.		X	X	X	X	X	
			Project Scoping - Identify Stakeholder: Define who is going to be involved during the project beyond the team and the three main roles.			X	X	X	X	
			Project Scoping - Evaluate BPM Maturity: Evaluate what is the maturity of the business process.			X	X	X		
			Project Kick-Off - Define sprint length: Define what would be the sprint length in week's terms. Every single sprint is going to have this duration.		X	X	X	X	X	

		Define tasks, Implement requirements, Get stakeholder feedback. Control project progress, Run retrospective], Release Sprint [Append Release Notes, Train IT operations and end users, Integration tests, Finish Documentation.]}	**Project Kick-Off - Create initial release plan**: Define what is going to be the plan for releasing the increment after every single sprint length.		X	X	X	X	X	
			Project Kick-Off - Establish architecture vision: Define the vision of the needed architecture for the project.			X	X	X	X	
			Project Kick-Off - Build team: Build the cross-functional team		X	X	X	X	X	
			Sprint 0 - Define Definition of Done & Definition of Ready: Create the Definition of Done and Ready. The definition of Done is all the parameters needed to accept the tasks as completed. The definition of Ready is the list of parameters that need to be met for considering a task as ready to be developed.		X	X	X	X	X	X
			Sprint 0 - Identify initial requirements: Define the initial requirements to launch the project.		X	X	X	X	X	
			Sprint 0 - Get stakeholder feedback: To have any feedback for the people involved in the project.		X	X	X	X	X	
			Sprint 0 - Control project progress: Define the progress of the project until now.			X	X	X		
			Sprint 0 - Run retrospective: Know what is working fine, what is not working, and what could be improved in the team.		X	X	X	X	X	
			Sprint 1-n - Refine process backlog: Refine the backlog with all the tasks with the needed information.		X	X	X	X	X	

			Sprint 1-n - Plan sprint, Define tasks: Define every single task that provide value to the project.		X	X	X	X	X	
			Sprint 1-n - Implement requirements: Develop every single user story.		X	X	X	X	X	X
			Sprint 1-n - Get stakeholder feedback: Get the feedback of the customers when the tasks are completed.		X	X	X	X	X	
			Sprint 1-n - Control project progress: Know What is the progress of the project? What is the increment of this sprint?			X		X	X	
			Sprint 1-n - Run retrospective: Run a Scrum retrospective when the sprint is over.		X	X	X	X	X	
			Release Sprint - Append Release Notes: Create the release notes when a new increment is built.		X	X	X	X	X	
			Release Sprint - Train IT operations and end users: Train the final users if needed.			X	X	X	X	
			Release Sprint - Integration tests: Create test cases that cover the functionality of the development.		X	X	X	X	X	
			Release Sprint - Finish Documentation: Create the final documentation for the increment.		X	X	X	X	X	X
DC.11 ABPM Phases	DTS.4 ABPM (Rosing and Gill, 2015)	{Agile Analysis, Agile Planning, Agile build, testing, and deployment}	**Agile Analysis**: Do all the analysis before starting the project.		X	X	X	X		
			Agile Planning: Elaborate the plan to develop the project.		X	X	X	X	X	
			Agile build, testing, and deployment: Iteratively develop all agile activities.		X	X	X	X	X	X

DC.12 ABPM Activities	DTS.4 ABPM (Rosing and Gill, 2015)	{Agile Analysis [High-Level Business Requirements], Agile Planning [High-level project plan], Agile build, testing, and deployment [Defining the Sprint Backlog, Sprint Planning, Performing Sprint, Testing, Demo Increment, Client Feedback Meeting, Retrospective, Deploying Increment]}	**Agile Analysis- High Level Business Requirements**: Create the requirements for the project.		X	X	X	X	X	
			Agile Planning - High-level project plan: Define the high-level plan for the project.		X	X	X	X		
			Agile build, testing, and deployment - Defining the Sprint Backlog: Take the most important requirements and put them in the sprint backlog.		X	X	X	X	X	
			Agile build, testing, and deployment - Sprint Planning: Define the plan for the sprint and all the requirements that need to be done.		X	X	X	X	X	
			Agile build, testing, and deployment - Performing Sprint: Develop the requirements, and conducting the Dayli meeting.		X	X	X	X	X	
			Agile build, testing, and deployment - Testing: Test every single requirement that is developed during the sprint.		X	X	X	X	X	X
			Agile build, testing, and deployment - Demo Increment: Demo the increment to the Product Owner and the stakeholders.		X	X	X	X	X	
			Agile build, testing, and deployment - Client Feedback Meeting: Get any feedback provided by the stakeholders during the demo.		X	X	X	X	X	
			Agile build, testing, and deployment - Retrospective: When the sprint ends a retrospective meeting is conducted so that the team can be better for the next sprint.		X	X	X	X	X	

			Agile build, testing, and deployment - Deploying Increment: Deploy the increment at the end of the sprint with a working product with all requirements done during the Sprint plus past requirements.		X	X	X	X	X	

Table 48 - Artifacts for Desing Components first and second iterations.

Design Component	**Source**	**Name**	**Why this could be helpful**	**SDLC that is also using it**				**Iteration**		
				DTS. 1	**DTS. 2**	**DTS. 3**	**DTS. 4**	**1**	**2**	**3**
DC.3 The BPM Lifecycle Artifacts	DTS.1 The BPM Lifecycle (Dumas et al., 2018)	{Process Identification [Process architecture of the selected process], Process Discovery [As-is business process model]}	**Process Identification - Process architecture of the selected process**: The final document of the architecture of the project.	X				X	X	X
			Process Identification - As-is business process model: The current state of the business process.	X	X		X	X	X	
DC.10 APBPM Artifacts	DTS.3 APBPM (Thiemich and Puhlmann, 2013)	{Project Scoping [Project Idea, List of Stakeholder], Project Kick-Off [Architecture Vision, SOA-MAP, First Release plan, Skill matrix], Sprint 0 [Def. of Done, Def. of Ready, Process Backlog, Story Map], Sprint 1-n [Sprint Backlog, Process Increment, Story Map], Release Sprint	**Project Scoping - Project Idea**: A document that clearly defines the project idea.		X	X	X	X	X	X
			Project Scoping - List of Stakeholders: A document having a list of all stakeholders of the project.		X	X	SS	X	X	X
			Project Scoping - Architecture Vision: A document with the vision of the architecture of the project.		X			X	X	X
			Project Scoping - SOA-MAP: A map with the services needed for the project.		X	X	X	X		
			Project Scoping - First Release Plan: A document that details the release plan for the project.		X	X	X	X	X	X

		[Training documents, Release Notes, Documentation]}	**Project Scoping - Skill matrix**: The skills that the development team needs to have to complete the project.		X			X		
			Sprint 0 - Def. of Done: A list of parameters that tasks need to be met for considering tasks as done.		X	X	X	X	X	X
			Sprint 0 - Def. of Ready: A list of parameters that tasks need to be met for consideration as ready for development.		X	X	X	X	X	X
			Sprint 0 - Process Backlog: The backlog of tasks to be developed.		X	X	X	X	X	X
			Sprint 0 - Story Map: A board that shows all the stories, their status, and who is working on them.		X	X	X	X	X	X
			Sprint 1-n - Sprint Backlog: The list of tasks to be developed during the sprint.		X	X	X	X	X	X
			Sprint 1-n - Process Increment: The result of merging newly developed stories with the past increment.		X	X	X	X	X	X
			Sprint 1-n - Story Map: A board that shows all the stories, their status, and who is working on them.		X	X	X	X	X	X
			Release Sprint - Training documents: Needed documents for training.		X	X	X	X		
			Release Sprint - Release Notes: A document with the final release notes after the increment is done.		X	X	X	X	X	
			Release Sprint - Documentation: A document with the final results of sprint review, and sprint retrospective.		X	X	X	X	X	X
DC.13 ABPM Artifacts	DTS.4 ABPM (Rosing and Gill, 2015)	{Agile Analysis [Selected business process and sub-processes, High-level user stories, Table of	**Agile Analysis - Selected business process and sub-processes**: A document with all the business processes and sub-processes selected to work on.			X	X	X		

		priorities and estimations], Agile Planning [Project plan], Agile build, testing, and deployment, Agile build, testing, and deployment [Sprint Backlog, Sprint Task Plan, Tests, Increment, Integrated Release]}	**Agile Analysis - High-level user stories**: A list of high-level user stories.		X	X	X	X	X	
			Agile Analysis - Table of priorities and estimations: A board with user stories estimated and prioritized		X	X	X	X	X	
			Agile Planning - Project plan: A document with a detailed project plan.		X	X	X	X		
			Agile Planning - Sprint Backlog: A list of stories to work on during the sprint.		X	X	X	X	X	
			Agile Planning - Sprint Task Plan: A document with a detailed test plan for the stories and the increment.				X	X		
			Agile Planning - Tests: Test cases to be performed on the stories.		X	X	X	X	X	
			Agile Planning - Increment: A working product with all developed user stories.		X	X	X	X	X	
			Agile Planning - Integrated Release: The final release with the final increment.		X	X	X	X	X	X