International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-11 Issue-4, March 2022

A Review Analysis of Design Models for Mechatronic Product Design Perspectives

Jean Bosco Samon, Damasse Harold Fotsa Tchouazong

Abstract*: The design of a mechatronic system is considered according to the concurrent engineering approach within the framework of a development cycle, is a methodological approach to master the design of complex systems and products. Competition on product quality and performance has become the main factor that companies face in today's market. These pressures are driving technological advances to provide systems with a constant increase in performance at reduced costs and in shorter delivery times. The performance includes better interoperability, improved reliability, good safety and smaller size that have determined the birth of mechatronic systems. Obtaining a product with these qualities in a reduced time would be beneficial for design companies if they follow an efficient design model appropriate for complex system. In this context, this paper conducts a discretized study on the main mechatronic design patterns in order to make their use understandable for the design of mechatronic products. Another essential aspect of the work lies on the comparison of the different existing design models analyzed on the basis of the multi-criteria analysis according to some fixed criteria in order to present a comparative study for choosing a better model for mechatronic product design perspectives.

***Keywords:* Design Process Models, Complex System, Mechatronic**

I. INTRODUCTION

The development (design) of a mechatronic system is considered according to the concurrent engineering approach within a development cycle. It is a methodological approach to master the design of complex systems and products [23]. The design of a mechatronic system must be carried out through an iterative approach consisting of a round trip between the functional definition of the product (customer's need and requirement) and its multi-physics definition. This design is presented in the form of a life cycle [1]. The life cycle of software, designates all the stages of the development of a software, from its conception to its end [2]. The objective of such a division is to define intermediate milestones allowing the validation of the software development, i.e. the conformity of the software with the expressed needs, and the verification of the development process, i.e. the adequacy of the implemented methods [3]. The origin of this division comes from the fact that errors are more costly when they are detected late in the development process [3]. The life cycle allows errors to be detected as early as possible and thus to control the quality of the software, the time required to complete it and the associated costs [4]. A number of research studies have established some life cycles (design patterns). In 1970, Winston Walker Royce developed the waterfall model, which is a linear business model that divides development processes into successive project phases and is commonly used in the construction industry [5]. From an idea, design activities proceed one after the other until completion. The waterfall model is a model that detects the anomaly late and is not useful for the design of complex systems which is sequential. The V-cycle is nowadays the best known and most used cycle in system design. It was developed in 1986 by Goldberg, and has its origin in software design to overcome the reactivity problems of the waterfall model [6]. Indeed, the V-cycle makes it possible to detect possible anomalies very early and thus to limit the return to the previous phases. It is composed of two parts: a top-down part (Top-Down) similar to the waterfall development allowing the decomposition of the project and a bottom-up part (Bottom-Up) which allows to make the integration and tests [3]. There are times when iterations are necessary, which leads to a long design time and therefore a loss in cost.

The b-design cycle is a reformulation of the waterfall model that was developed in 1988 by Birrell and Ould. This cycle model was developed based on the observation that complex systems require a lot of maintenance [7]. Birrell and Ould extended the operational life cycle (referred to as the maintenance cycle) and then, by attaching it to the waterfall model, developed the b-model. The latter covers about two-thirds of the life cycle model [5]. Like its waterfall cousin, the b-model detects errors late, making the design process iterative through the reuse of tests by the maintenance block.

After presenting his method for developing complex applications for the first time in 1986, the American software engineer Barry W. Boehm published his model in "A Spiral Model of Software Development and Enhancement" in 1988, and in an equally broad context [8]. In this publication, he describes the spiral model as a possible alternative to the previously established waterfall model and at the same time serves as an experimental basis. In contrast to the waterfall model, the spiral model does not assume that the tasks of software development should be organized in a linear way, but in an iterative way which leads to a long design time.

Manuscript received on January 14, 2022.
Revised Manuscript received on March 12, 2022.
Manuscript published on March 30, 2022.
* Correspondence Author
Jean Bosco Samon*, Department of Mechanical, National School of Agro Industrial Sciences, University of Ngaoundere, Ngaoundere, Cameroon. Email: jboscosamon@gmail.com
Damasse Harold Fotsa Tchouazong, Department of Mechanical, National School of Agro Industrial Sciences, University of Ngaoundere, Ngaoundere, Cameroon. Email: damasseharold31@gmail.com

The Y-cycle or also called 2TUP (Two Tracks Unified Process) proposed more recently by the company VALTECH in the early 2000s, dissociates the technical aspects from the functional aspects before starting the realization [10]. The Y model is a design model that is also suitable for the design of complex systems, but its iterative nature leads to a long design time, resulting in cost losses. In today's market, competition over product quality and performance has become the primary factor facing companies. These pressures are driving technological advancements to provide systems with an ever-increasing performance at reduced costs and shorter delivery times [11]. These demands drive improvements in system functionality, such as better interoperability, increased performance, improved reliability, good safety, and a smaller size that have determined the birth of mechatronic systems [12]. In this context, this chapter aims to make an analysis of the main existing design models and to present comparative tables which will take into account the design capability of the complex systems of the different models in order to choose the best mechatronic design model.

II. THE MAIN DESIGN CYCLES

Based on ISO/IEEE 15288:2001 (ISO 2001) Many research works are located in the design of systems integrating the pre-design phase ("Concept stage") and the development phase [14]. We have represented these different stages in the illustration below as depicted in figure 1:

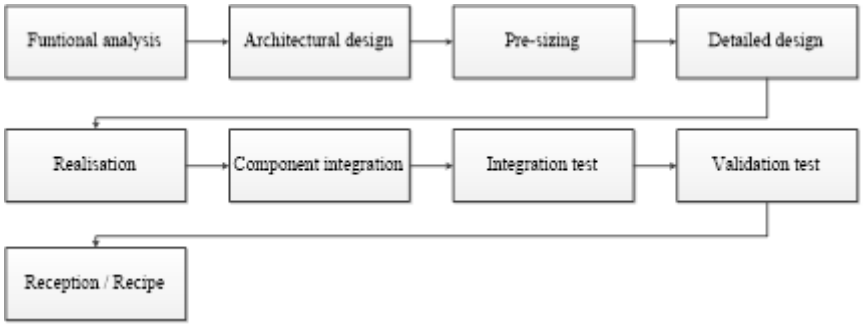

Fig. 1. Representation of the different stages of the design phase [14]

Based on the synthesis of [10] and some previous work, the following is a presentation of the main mechatronic design cycles.

A. The waterfall design model

— Operation of the cascade model

The development of the classic waterfall model is credited to computer scientist Winston Walker Royce. However, Royce did not invent it. His 1970 essay "Managing the Development of Large Software Systems" is more of a critical analysis of linear models. As an alternative, Royce proposed an iterative, incremental model in which each phase would build on the previous one and verify the results [9].

He recommended a seven-phase model that proceeded in several stages (iterations): system requirements, software requirements, analysis, design, implementation, testing, and operation.The so-called waterfall management model (figure 2) is based on the phases defined by Royce but provides for a single iteration.

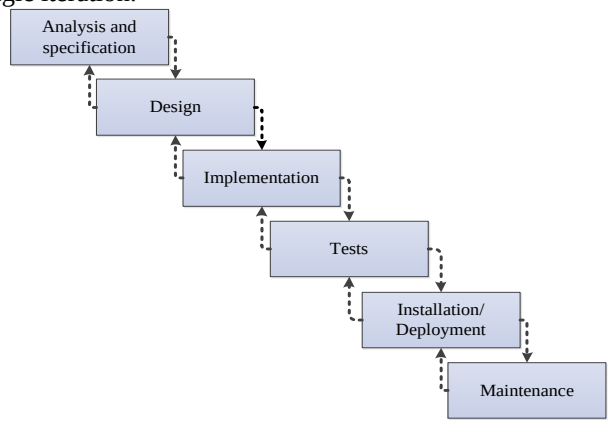

Fig. 2. Cascade Model [13]

In practice, several versions of the waterfall model are used. The most common models divide the development process into five phases. Phases 1, 2, and 3 as defined by Royce are sometimes combined into a single phase, referred to as the requirements analysis [10]. Analysis (planning, requirements analysis and specification), design (system design and specification), implementation (programming and testing of modules); testing (system integration, system and integration testing) and operation (delivery, maintenance, enhancement).

— The phases of the cascade model

In the waterfall model, the different phases of a development process follow each other. Each phase ends with an intermediate result (step). These different phases are [13]:

- Analysis

Every software project starts with an analysis phase including a feasibility study and a requirements definition. The costs, performance and feasibility of the software project are estimated during the feasibility study. The feasibility study is used to create a specification (a rough description of the requirements), a project plan, a project budget and, if necessary, a quotation for the client.

- Design

The design phase is used to develop a concrete solution concept based on the requirements, tasks and strategies determined in advance. In this phase, the developers develop the software architecture and a detailed software design plan and thus focus on concrete elements such as interfaces, frameworks or libraries. The result of the design phase includes a design document with a software build plan as well as test plans for the individual elements.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

➢ Implementation

The software architecture developed during the design phase is implemented during the implementation phase, which includes software programming, troubleshooting and module testing. In the implementation phase, the software project is translated into the desired programming language.

The individual software components are developed separately, checked in module tests and integrated step by step into the overall product. The result of the implementation phase is a software product that is tested for the first time as an overall product in the next phase (alpha testing).

➢ Test

The test phase includes the integration of the software into the desired target environment. As a rule, software products are first delivered to a selection of end-users in the form of a beta version (beta tests). The acceptance tests developed in the analysis phase are used to determine whether the software product meets the previously defined requirements. A software product that has successfully passed the beta tests is ready for release.

After successful completion of the testing phase, the software is released to production for operation. The final phase of the waterfall model includes delivery, maintenance and enhancement of the software.

B. The V-shaped design model

— How the V-model works

The V-cycle is nowadays the best known and most used cycle in system design. It was developed in 1986 by Goldberg, and has its origin in software design in order to overcome the reactivity problems of the waterfall model [6]. Indeed, the V-cycle makes it possible to detect possible anomalies very early and thus to limit the return to the previous phases. It is composed of two parts: a top-down part (Top-Down) similar to the waterfall development allowing the decomposition of the project and a bottom-up part (Bottom-Up) which allows the integration and testing [3].

The uplink allows analysis and verification before the integration of the components: it sends the specifications of the downlinked part to the phases opposite the uplinked part. In the same way, the phases of the rising part report the detected defects/deviations to the specifications for each phase of the falling part (figure 3). The vertical axis represents the level of detail and the horizontal axis represents the progress over time.

The structure of the V-cycle allows to compare two to two steps of its two main parts (Functional specification / Functional validation, System specification / System validation, Subsystem definition / Subsystem validation and Component definition / Component validation) see figure 3. It allows to go from the most general to the most detailed during the realization, then from the most detailed to the most general during the validation.

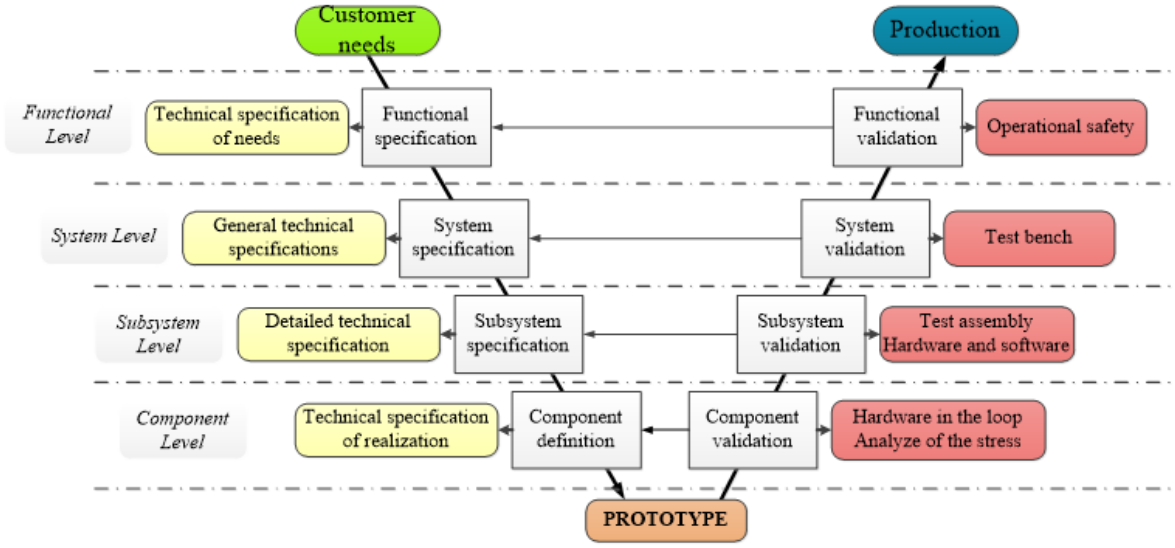

Fig. 3. V-model for the design of mechatronic systems [15]

— The different phases of the V-cycle

Mihalache breaks down the V-cycle into five (5) parts: analysis/specification and design for the top-down part, verification and validation for the bottom-up part, and implementation between the two parts [11]. In more detail, Jardin breaks down the V-cycle into nine (9) parts: functional specification, system specification, subsystem definition and component definition for the top-down part, functional validation, system validation, subsystem validation, and component validation for the bottom-up part and prototype realization between the two [16]. So, according to Jardin, the design tasks of mechatronic systems are divided into four parts or levels.

➢ The functional level

It corresponds to the definition of the functionalities, interfaces, constraints and requirements of the system as well as the preparation of the quality plan, the validation plan and the feasibility study. The definition of behavioral performance constraints (reliability, maintainability, safety, availability, recyclability) is also carried out during this phase [11]. In the same way, the information relative to the size, the environmental impact, the cost and the deadlines are taken into account. This is the phase during which the expectations expressed by the customers are translated into technical specifications [16].

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

➢ The system level

It corresponds to the definition of the system architecture. It is during this phase that the choice of one or more subsystems allows realizing each functionality is made. The interactions that these different subsystems should have are also proposed. At the end of this stage, the general technical specifications of the product are described [17].

➢ The subsystem level

It is at this level that the subsystems selected in the previous level are in turn described. Each subsystem can be decomposed in turn into subsystems down to the elementary components (parts). Again, within each subsystem, the interactions between its components are defined. At the end of this phase, the detailed technical specifications are described [11].

➢ The component level

This is the last phase of the top-down part of the V-cycle. This level then completes the design of the product. The components are described in detail and the technical specifications for realization are given. At the end of this last phase, virtual prototyping must be possible. This prototype allows to test, optimize and validate the system without using real prototypes [17].

C. The b-model

— Operation of the model in b

The basic principle is the reuse of test scenarios established during the development phase (operational procedures, non-regression tests) [5]. This is where its main interest lies: during the project phase, the reuse of tests and procedures for putting into operation is planned. The extension of the maintenance cycle was done to ensure that the continuous improvement of the software or system would become part of the development stages. In addition, they felt that an alternative to obsolescence had to be found so that improved or even newer systems could be developed as spin-offs of the original system. In this sense, the b-model (figure 4) was an attempt to modify the waterfall model by creating an evolutionary improvement process that was captured by the spiral model. The b-model, however, is more suitable for small project development like its cousin, the waterfall model [8].

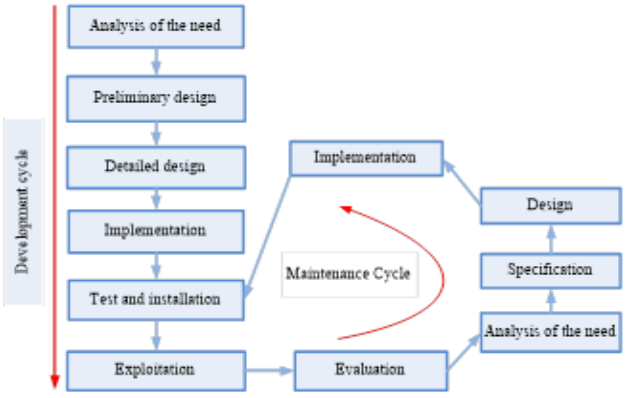

Fig. 4. Design model in b [5]

— The different phases of the model in b

The b-model is a design model derived from the waterfall model, consisting of two cycles that are the development cycle and the maintenance cycle.

Development cycle

In this branch of the development cycle, we have the following phases which have the task of specific activities.

➢ Needs analysis

The first phase of the model developed by Birrell and Ould allows for [18]::

- Preliminary study: In this first task, we have the overall definition of the problem (context), the evaluation of possible strategies, and the evaluation of resources, costs, and deadlines. The preliminary study shows the analysis report.
- Expression and analysis of needs: Here we have the subtasks like expected functional requirements (operations, etc.), and expected non-functional qualities (performance, portability, etc.).

After the expression and analysis of requirements, specifications, analysis document (specification) and system test plan are obtained.

➢ Preliminary design

Preliminary design is above all a matter of organization; it relies on the functional and structural specification points of view of the analysis, but also on the frameworks of the technical design. It ends when the design is organized as follows [19]: its target deployment, its operating model and its logic model. It should be move from object analysis to design, integrate the system's business and application functions into the technical architecture and adapt the generic design to the specifications provided by the analysis.

The preliminary design corresponds to the decomposition and organization of the application into simpler modules defined by an interface. For example: database, operating environment, interface [13].

➢ Detailed design

The detailed design is an activity that is part of the organization defined by the preliminary design. The logic model is particularly important in this phase, as the largest volume of information is generated in detailed design. It is therefore possible to assign categories to different people, who can work independently of each other. Detailed design is therefore based on design categories organized according to both technical frameworks and business-specific groupings. The designers then build the classes, human-machine interface (HMI) views, interfaces, tables and methods that will give a "ready-to-code" image of the solution [19].

It allows to describe, for each module, the way services and functions are realized by relying on the essential algorithms, the data structures used,... [13]. Finally, the content of the subsystems must be specified in order to complete the software configuration. The level of abstraction targeted by the detailed design stage is the design of the components. The aim is to have the most precise idea possible for the manufacture and assembly of the software configuration subsystems [19]. At the end of this phase, the design document (specification), the prototype, the global test plan, the test plan per module.

➢ Implementation

It is the translation of the design into a programming language or implementation using development tools, building the software components. The objective is to prepare the programming files, commented source code, and the prototype (Olivier, 2007).

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

➢ Test and installation

The purpose of this phase is to verify and validate by asking questions such as: Is it well done? Does the program meet the specification?

So we have software (i.e. computer system) verification, software validation, audit of the information system, and validation of the information system.

The verification and validation tasks allow to [20] :

- Verification: to see if the product being developed meets the definition of needs? (is it the product?);
- Validation: to see if the product under development fulfills the functionalities desired by the user? (is it the right product?).
- Unit tests, integration tests and system tests are performed to produce a verification report by test [18].

The facility corresponds to putting into operational use at the users' premises and Data conversion [13], sometimes restricted to selected users.

➢ Operation

The operation determines the use of the software once installed (and for which the recipe is made).

Maintenance cycle

The maintenance cycle evaluates the development cycle in order to consider the reuse of tests and procedures for putting the system into operation. The main tasks related to it are [13] corrective maintenance to correct errors, adaptive maintenance that allows for adaptation to changes in the environment, perfective maintenance for improvements, and preventive maintenance to facilitate future maintenance operations.

D. The spiral design model

— Operation of the spiral model

Problems arising in the software development process can have a wide range of consequences on the project as a whole. In all cases, increased costs, additional effort, and a delay in the release of the software are to be expected, factors that can quickly turn into an existential problem, especially for small publishers [8]. With its incremental and iterative approach that also includes regular risk assessment in the form of draft prototypes, analyses or simulations, the spiral model (figure 5) is supposed to avoid such scenarios or at least mitigate their negative impact.

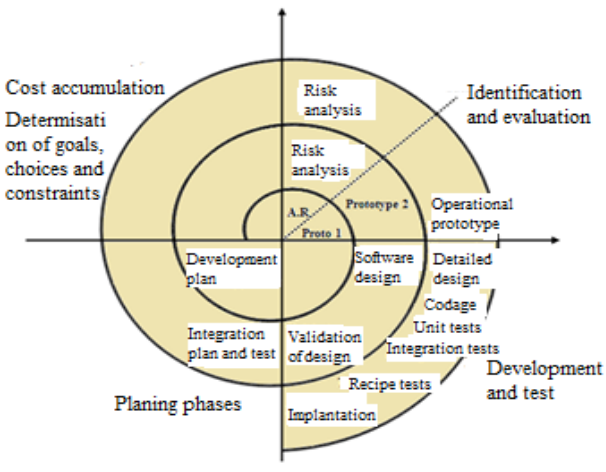

Fig. 5. Spiral Design Cycle [21]

— The phases of the spiral model

➢ Phase 1: Definition of objectives and alternatives

A typical cycle in the spiral model begins with the determination of the objectives to be associated with the individual steps in the development process. This could be, for example, to improve performance or to extend functionality. At the same time, alternatives for the implementation (e.g. design A vs. design B) must be defined and the general framework as well as the necessary costs or work time must be determined.

➢ Phase 2: Review of alternatives

The next step is the evaluation of alternatives, in which the objectives and the general framework serve as reference values. In this phase of the spiral model cycle, the aim is to identify areas of uncertainty, i.e. those areas of the project that carry a significant risk for the progress of the development project. This is followed by the development of the least risky and most cost-effective strategy**,** where methods such as prototyping, simulations, benchmarking, analysis models, and user surveys can be employed.

➢ Phase 3: Development and control of the intermediate state

After the risk analysis, the actual software development phase begins, which is always characterized by relative residual risks. If the development process is affected by performance or user interface risks, or by risks concerning the control of internal interfaces, an evolutionary development strategy is first possible, in which the project is specified more precisely and prototypes are optimized. The code is written and tested several times until the desired result is achieved, which then serves as a low-risk basis for further development steps.

➢ Phase 4: Planning the next cycle

With the end of a cycle, the planning of the next cycle already begins. This can be the regular progress of the project if the objective of the cycle has been reached and the next objective has to be defined. But it can also be about finding solutions if the previous development stage did not go as planned. For example, the previous strategy can be replaced by one of the previously defined alternatives or by a new alternative. With this alternative, it is then possible to start a new attempt to reach the goal.

E. The Y design cycle

— Operation of the Y-model

The Y-cycle or also called 2TUP (Two Tracks Unified Process) proposed more recently by the company VALTECH in the early 2000s, dissociates the technical aspects from the functional aspects [10]. The process is based on three phases (Fig. 6): A functional branch, a technical branch, and a branch of realization. The first two are addressed simultaneously during the requirements capture phase and the analysis phase. The last branch consists in bringing together the two previous branches, allowing to lead the realization and the delivery of the system (or product). In this cycle, the evolution of time is represented along the vertical axis.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

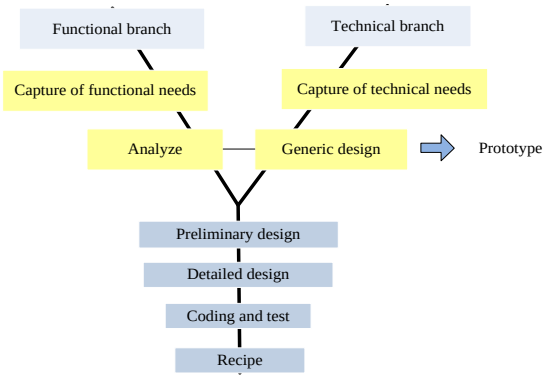

Fig. 6. The 2TUP model [5]

The Y design model is an iterative model based on the following operating structure [19] :

- Iteration 1: (develop the system proof of concept functions and integrate the planned development tools) ;
- Iteration 2: (focused on the architecture and the technical realization prototype) ;
- Iteration 3: (implement the highest priority functions in order to present a first deployment version to the users, improve and complete the technical architecture) ;
- Iteration 4...n: (perform functions until the initially planned system is complete).

— **The different phases of the Y model**

The 2TUP model as we had presented above, consists of three branches that are used during the design process to [22]:

➢ Functional branch

- Functional requirements capture: focuses on the user's business;
- Analysis: study precisely the functional specification in order to get an idea of what the system does without worrying about the technologies to be used.

➢ Technological branch

- The capture of the technical needs: See all the constraints and the choices dimensioning the design of the system (tools + material + integration constraint) this phase also allows to make the capture of the technical needs with the existing;
- Generic design: defines the components needed to build the technical architecture.

➢ Realization branch

- Preliminary design: allows the integration of the two previous branches, i.e. the technological branch and the functional branch;
- Detailed design: this phase aims at studying the realization of each component ;
- Coding and testing allows to code the components + test them;
- Recipe: in this last phase, it is a question of validating and selling the product

III. ADVANTAGES AND DISADVANTAGES OF THE MODELS

Table 1 shows us some key advantages and disadvantages of the main design models presented above.

Table-I: Advantages and Disadvantages of The Models.

Models	Benefits	Disadvantages
Cascade model	• Simple to use; • Clearly defined steps ; • Cost estimates at the beginning of the project	- Not suitable for complex projects; - Little room for adjustment ; - Errors are sometimes only detected at the end.
V-shaped model	• Controlled risks and better planning ; • Quality improvement; • A transparent process for the entire product life cycle.	- Lack of flexibility (makes it difficult to use) ; - Difficulty in getting feedback on modifications.
Model in b	• Its simplicity; • It foresees, during the project phase, the reuse of the tests and the procedures of exploitation thanks to the maintenance cycle.	Requires a long time to design; Difficulty in getting feedback for modifications; - Late detection of anomalies.
Spiral Model	• Periodic control due to risks ; • Perfect coordination between technical requirements and design; • Maximum control of costs, resources and quality of the product project; • Suitable for innovative technical environments.	- High management effort (its complexity) ; - Regular decisions can delay; - Because of the subdivision of the development process, design errors and inconsistencies can easily find their way into the final product; - Unsuitable for small projects with reasonable risks.
Y model	• Focus on technology and risk management; • Defines the profiles of the participants, the deliverables, the schedules, the prototypes.	- Superficial on the upstream and downstream phases of development; - Does not offer standard documents; - Requires several iterations to design, which takes a long time.

IV. COMPARISON OF MODELS

The comparison will be done on some fixed criteria in order to be able to output the comparative table for the selection of the model which will best suit the design of the mechatronic systems.
In Table 2, we denote with a "+" the fact that the model meets the assigned criterion and with a "-" the opposite.

For the elaboration of the table 2 which compares the main existing Mechatronic design models we proceeded with the multi-criteria analysis by fixing the criteria before proceeding to an assessment according to each criterion for each model. The assessment leads us to choose the V-model which is still more or less the best model for the design of complex (mechatronic) systems until now despite some shortcomings.

Table- II: Comparison of Models.

	Models / Criteria	Cascade model	V-shaped model	Model in b	Spiral model	Y model
1	Ease of use: complexity of control for the designer	+	+	-	-	+
2	Independence between phases: linearity	+	+	+	+	+
3	Changes to previous phases: consequence of returning to a step	-	+	+	+	-
4	Ability to manage risks during the design process: possibility to avoid iterations	-	+	+	+	+
5	Adaptation from a major project: the duration of the project	-	+	-	+	-
6	Ability to design mechatronic systems: non-sequential	-	+	+	+	+
7	Iterative nature: the repetition of the design process	+	+	+	+	+

V. CONCLUSION

The design of mechatronic systems is particularly long and difficult, because of their strong functional integration, their multi-domain and multi-physics aspects on the one hand, and the resulting couplings on the other hand. At each stage of the design cycle, at all levels of abstraction or decomposition of the system, the designer must be able to study the trade-offs and justify the design choices based on the analyses of the various disciplines, while guaranteeing the coherence of the solution. In this paper we had made a characteristic study of the main existing mechatronic design models before conducting a comparative study on the criteria for the choice of the best appreciated design model for the design of complex (mechatronic) systems. The comparison made us choose the V-model as the best model for the design of mechatronic systems today. On the other hand it is not exempted as it has some disadvantages which should be corrected, hence the focus of future research work.

REFERENCES

1. AFNOR-XP. XP E 01-013 - Mechatronics – Product life cycle and design. 2009.
2. Ruparelia N. B. Software Development Lifecycle Models. Hewlett-Packard Enterprise Services, 2010. 35(3).
3. Petit F. Life cycle of computer systems. 2007.
4. Hugues A.-M. Software engineering. 2002.
5. Guizani A. Multi-agent approach for the optimal design of mechatronic systems. Université Paris-Saclay; Mechanics, Modeling Research Laboratory ... 2016.
6. Hammadi M. Contribution to the integration of multi-physics modeling and simulation for mechatronic systems design. École Centrale Paris. 2012.
7. Birrell N. D., and Ould M. A. A practical handbook for software development: Cambridge University Press. 1988.
8. Vanshika R. Software Development Life Cycle Models Comparison, Consequences. International Journal of Computer Science and Information Technologies, 2015. 6(1), 16-17.
9. Royce W. W. Managing the development of large software systems. In proceedings IEEE WESCON Los Angeles, 1970. 26.
10. Tahan M., Amara T., Jean V., and Philippe L. PThe X development method, another perspective on the life cycle. 2011.
11. Mihalache A. G. Modeling and reliability assessment of mechatronic systems: application on embedded system. University of Angers. 2007.
12. Giacomo B., Cesare F., and Roberto B. A model-based design methodology for the development of mechatronic systems. Elsevier. 2014.
13. Dufour B. Software engineering: Development process. Université de Montréal. 2010.
14. Warniez A. Integration metrics for architecture selection in mechatronic system design. École Centrale Paris (ED287). 2015.
15. Casner D., Houssin R., Knittel D., and Renaud J. An approach to design and optimization of mechatronic systems from multidisciplinary optimization and based on feedback. Paper presented at the Congrès Français de Mécanique 2013. 2013.
16. Jardin A. Contribution to a sizing methodology for mechatronic systems: structural analysis and coupling to dynamic optimization. INSA de Lyon. 2010.
17. Touron M., Gomand J., Dieulot J.-Y., and Barre P.-J. Architectural design of a mechatronic operator assistance system using Bond-Graph. 2012.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

18. Vachon J. IFT6803: Electronic commerce software engineering. 2003.
19. Roques P., and Vallée F. Software architecture, UML 2 in action: From requirements analysis to design. 2007.
20. Olivier G. Course on Analysis and Design of Information Systems (Tools and Models for Software Engineering). University of Bordeaux 1. 2007.
21. Boehm B. W. A spiral model of software development and enhancement. Computer. 1988. 21(5), 61-72.
22. Barthon S. 2TUP Conference. 2012.
23. Boucerredj L. Safety of Operation: Search of Critical Scenarios in Mechatronic Systems. UNIVERSITE BADJI MOKHTAR ANNABA. 2015.

AUTHORS PROFILE

Jean Bosco Samon Graduated, with an Industrial Engineering and Maintenance bachelor's degree from Institut Technology of Douala-Cameroun, the Mechanical Engineering and Industrial automation Master degree, and with a Ph.D. in Mechanical Engineering in the field of Equipment Engineering and Industrial automation from the National Advanced School of Agro-Industrial Sciences of Ngaoundere, Cameroon. His research interest is mechanical and mechatronical design with application in complex systems. He is also a coordinator of the laboratory of Mechanic, Materials, and Photonic. Jean Bosco Samon is the corresponding author and can be contacted at: jboscosamon@yahoo.fr or bosco.samon@univ-ndere.cm

Damasse Harold Tchouazong Fotsa, received the Mechanical Engineering and Production bachelor's degree from Institut Technology of Ngaoundere-Cameroun, the Mechanical Engineering and Industrial automation Master degree from National Advanced School of Agro-Industrial Sciences of Ngaoundere, Cameroon, in 2018 and 2021 respectively. He is currently a Ph.D. student at the same school in the Doctoral field of Equipment Engineering and Industrial automation. His research interests include is topological mechatronic. Damasse Harold TCHOUAZONG FOTSA can be contacted at: damasseharold31@gmail.com