COMPARATIVE ANALYSIS OF SOFTWARE DEVELOPMENT LIFE CYCLE MODELS:

EVOLUTIONARY AND COMPONENT BASED DEVELOPMENT.

BY

OLADITI, Adewale John

125/19/1/0064

Software Engineering

**Abstract**

The software development models is one of the core factors that determines the quality of a software product and the success of development work. Without a suitable model put in place, the whole software development work will be unsuccessful, so software engineers have come up with different models and have also compared different models so as to determine the most suitable model for the software depending on some feature like cost, duration, project size, complexity, and risk. (Adeagbo et al., 2017) With all these core features, we’ve been able to predict the fate of a software project if we use a particular software development life cycle model, even before we embark on it. (Akinsola et al., 2021). Professionals in this field have done great jobs on comparing the traditional models like Waterfall, V-model and Spiral Model(Akinsola et al., 2021), but in this journal, we will be comparative one of the traditional models with other models, this article compares the Classical Waterfall model, Evolutionary model and Component-Based development using the key features of software development lifecycles mentioned above.

**Keywords: Waterfall model, Component Based Development, SDLC models, Evolutionary model.**

1. **Introduction**

Software engineering involves the systematic processes in the design, development and maintenance of quality software product which is achieved through a well-articulated SDLC model. Thorough analysis is required to deliver software product within a timeline and budget. The waterfall model, evolutionary and component based development will be looked into to determine the best fit for complex project, level of risk, duration of project, cost implications and other factors associated to the development of a quality product. (Adeagbo et al., 2017)(Akinsola et al., 2021)

The Software Development Life cycle (SDLC) involves eight phases which are planning, requirement gathering and analysis, system architecture and design, coding, integration, testing and debugging, deployment, maintenance, and without proper modeling of this developmental processes, our project is at risk of failure. So, this paper will discuss and compare a traditional waterfall model, evolutionary and component based development, so as to help software companies/engineers to determine the best fit/safe model to use in software development when this three models are the main options. (Adeagbo et al., 2017)(Akinsola et al., 2021)

1. **Evolutionary Model**

It is the union of iterative and incremental Software Development Life Cycle model. In this model, the system requirement is first decomposed into subsystems that can be incrementally built and delivered. At first, the system’s core module is being developed and then improved in the successive versions by adding new functionalities.

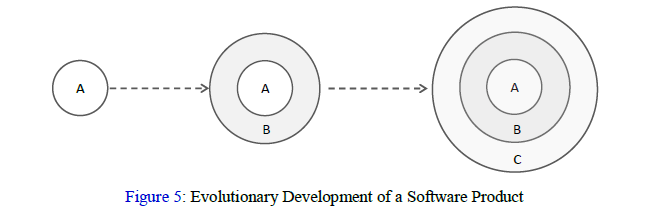
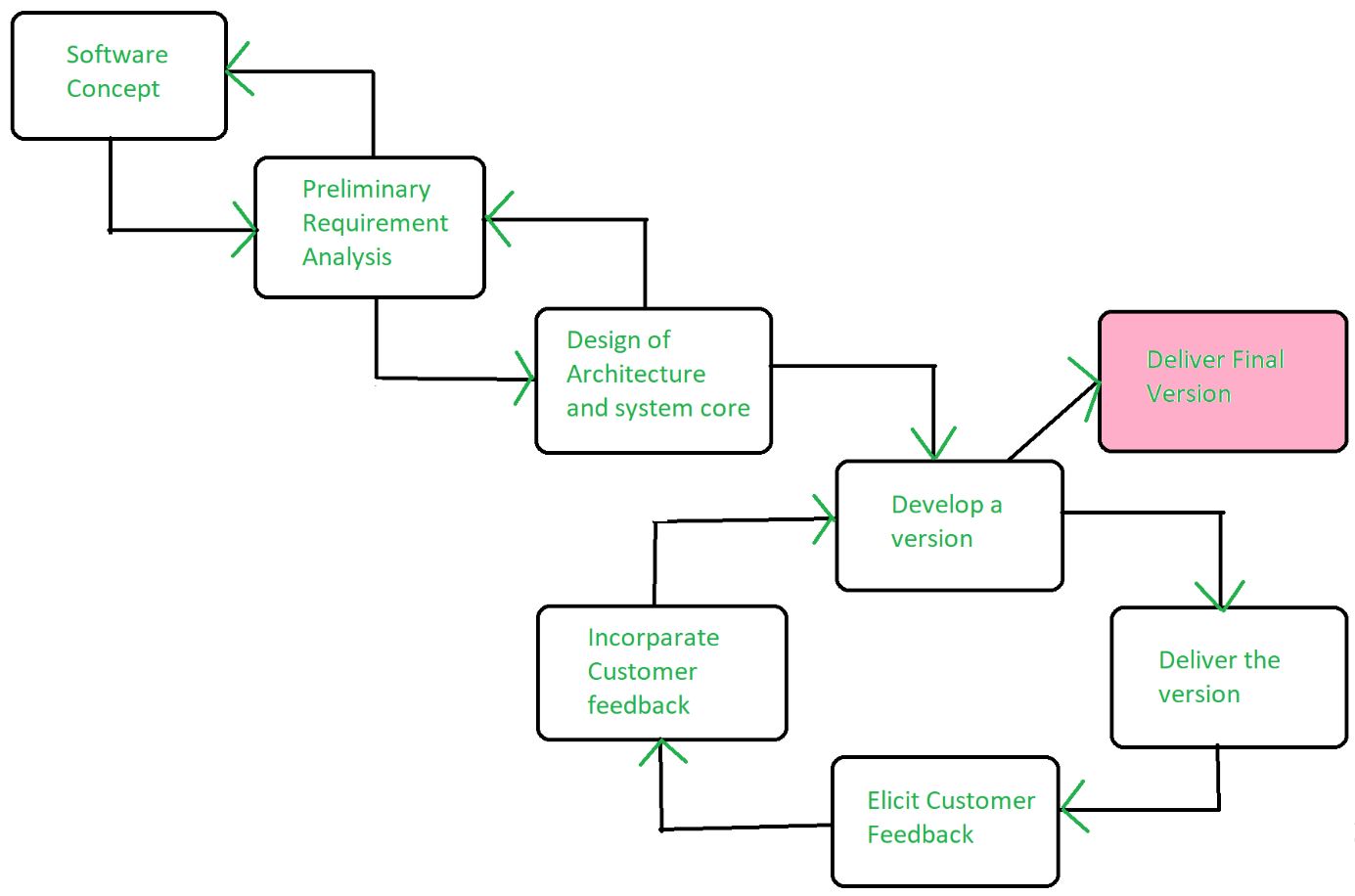
Figure 1. Evolutionary Development of Software Product. 

Figure 2. Evolutionary Model of Development.

Habitually, evolutionary models are used when clients want the product to be delivered by increments and can’t wait till the product is completed before making use of some functionalities.

* 1. **Merits of Evolutionary Model.**(Techopedia Inc, 2021)
* Project size: Evolutionary model is suitable for large projects like life and mission-critical products.
* First Version: the first version of the system is delivered with core functionalities before the delivery of complete version.
* Detailed requirements: It aids in precise drawing of user requirements during delivery of successive versions.
* Module Testing: Involves meticulous core module testing to reduce errors.
* It requires less resources for system development.
* Requirement: the requirement is always changing at each evolution.
* Team involvement: Involves all development teams.
* Risk evaluation: the risk evaluation for evolutionary model is low.
* Initial operating time is less.
  1. **Demerits of Evolutionary Model.**(Techopedia Inc, 2021)
* Project size: not suitable for small projects.
* Client satisfaction: it is hard to decompose system into versions that meets client satisfaction.
* Delivery difficulties: it is hard to decompose system into versions that can be incrementally delivered.
* Error detection: It may involve fast repair.
* Unceasing changes can ruin system structure.
* Undetectable process not well-supported by documentation.
* Compulsory special tools and techniques.
* Complex management.
* Skill requirement: Prerequisite high skilled resources for risk analysis.
* Progress of project depends on risk analysis phase.
  1. **Area of applications of Evolutionary Model.**

1. Scientometrics:  “quantitative study of science, communication in science, and science policy” (Hess, 1977, at p. 75).
2. Object oriented software development since the system can be easily portioned into units in terms of objects.
3. Artificial Intelligence.
4. Mission-Critical projects.
   1. **When to use Evolutionary models.**

* When customers want fast delivery of software product.
* When customer’s requirement keeps changing.
* In large system where finding incremental modules is easy
* When clients are in haste of using core features instead of waiting for complete product.
* When there is a prediction that customer might add additional functionality in future.

1. **Component Based Development Models**

The component-based development/software engineering (CBSE) uses object-oriented technologies which emphasis on classes as the entity that encapsulate data and algorithms. In component-based architecture, classes can be reused, multiple classes can be used and these classes are basically prepackaged components.(Techopedia Inc, 2021) This model uses various characteristics of spiral model and it is also evolutionary in nature. Components are more abstract than object classes and can be considered to be stand- alone service providers. (Engineering, 2004). It develops large complex software through cost-effective, fast and modular approach. CBSE is mainly based on the concept of reusability.

Component Based software development is an idea to develop system by selecting already developed components and assemble them into a new system. Various component technologies such as Component Object Request Broker Architecture (CORBA), Component Object Model (COM), and Enterprise JavaBeans (EJB) are used to develop the CBD system. Each plays its vital role regarding platform and language compatibility to make CBSE most suitable choice of most developers(Muhammad & Mahboob, 2016)

Object-oriented modelling consequences in a surfeit of fine-grained classes, objects and interactions. The idea behind CBD is to integrate the related parts and reuse them collectively. These integrated parts are known as components.

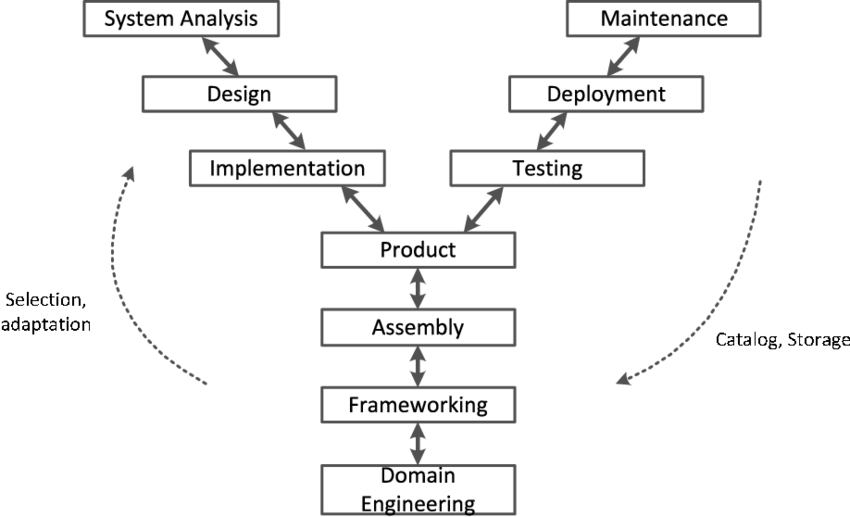


Figure 3. Y-process model for Component Based Development.

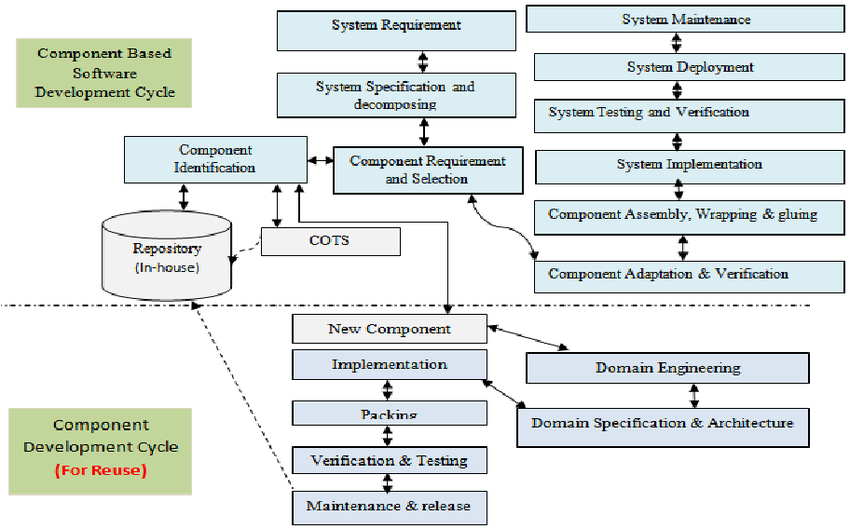


Figure 5. CBSE

(Khan et al., 2013)

* 1. **Design principles of CBSE.**(Engineering, 2004)

1. Independent component
2. Encapsulated component
3. Well-defined interface for Communication.
4. Shared component platform to reduce development costs.
   1. **Key goals of CBD**(Techopedia Inc, 2021)

* Project duration and cost: It saves time and money when developing large system.
* Software quality: It enhances the software quality.
* Reusability and flexibility: It increases reusability and flexibility.
* Early Error/defect detection: It trace and detects defect within the system early on and thus save cost and time in the development process.
* Delivery time: Minimized delivery time.
* Efficiency is improved since application development is concentrated on.
* Quality is improved since additional time can be permitted.
* Minimized expenditure.
  1. **Demerit of Component Based Development.**(Techopedia Inc, 2021)

1. Compromises in requirements are needed.
2. Less control over the system’s evolution.
3. Increasing complexity.
4. Change management and adaptation to the new approach.
5. Lack of a unique, widely accepted integrated approach to MBD.
6. Model-based testing.
   1. **The specific routines of CBD are;**(Techopedia Inc, 2021)

* Component development.
* Component publishing.
* Component lookup as well as retrieval.
* Component analysis.
* Component assembly.
  1. **Application of Component Based Development.**(Techopedia Inc, 2021)

1. API-driven applications.
2. Cloud-based application.
3. Service-oriented architectures.
4. Robotic systems.
5. Mission critical systems.
6. Medical.
   1. **When to use Component-based model.**

* There is greater difficulty in managing requirements;
* Greater efforts are needed to develop reusable units.
* Greater efforts are needed for providing component specifications and additional material that help developers/consumers of the components.

1. **Comparative analysis of Waterfall, Evolutionary and Component based development.**(Akinsola et al., 2021)

|  |  |  |  |
| --- | --- | --- | --- |
| **Features/models** | **Waterfall model** | **Evolutionary model** | **Component Based Development** |
| **Complex project** | Not suitable | Suitable | Very Suitable |
| **Project size** | Medium | Large | Medium to large |
| **Project Duration** | Long time duration | Shorter duration | Short time duration |
| **Planning period** | Requires long term planning | Requires less term planning | Requires medium term planning |
| **Project development method** | Top-down with sequential | Top-down and bottom-up with prototyping. | Top-down and bottom-up with parallel development |
| **Project with risk** | Not suitable | Suitable | Very suitable |
| **Implementation/initial cost** | Not costly | Costly | Less costly |
| **Reliability of the model** | High level of risk and insecurity | Reliable and secured | Reliable and secured |
| **Error discovery** | Quite late | Very Early | Early |
| **Associated cost** | medium | high | low |
| **Implementation** | Simple | Complex | Very complex |
| **Scalability** | Not suitable | Suitable | Very suitable |
| **Risk analysis** | No risk assurance | High risk assurance | Medium risk assurance |
| **Long term project** | Not suitable | suitable | Very suitable |
| **Maintenance** | Less maintainable |  |  |
| **Flexibility** | Not flexible | Less flexible | Flexible |
| **Initial requirement specification** | Well documented | Changes regularly | Changes regularly |
| **Requirement gathering** | Agreement between the client and engineers. | Disagreement between the client and engineers | Disagreement between the client and engineers |
| **Cost estimation** | Easy to estimate | Difficult to estimate | Difficult to estimate |
| **Testing** | At the middle | At the end of each phase | At the middle of each phase |
| **Guarantee of success** | Less | High | High |
| **Predictability of outcome** | Yes | No | No |
| **Documentation** | Easy to document | Very complex | Complex |
| **Coding** | Middle of projects |  | Less coding at middle of the project |
| **Model focus** | Quality control and regulation compliance | Fast delivery of quality product incrementally | Quality improvement of already existing software. |
| **Areas of application** | * Civil Engineering Construction, * Aircraft/Airplane Construction, * Banking, * Military, * Nuclear System, * Application Development | Same as water-fall. | * API-driven applications. * Cloud-based application. * Service-oriented architectures. * Robotic systems. * Mission critical systems. * Medical. |
| **When to use** | * Only if the requirements are very well known, clear and constant. * The definition of the product is stable. * It is understood technology. * Extensive resources with necessary expertise are freely available * • The project duration is short. | * When customers want fast delivery of software product. * When customer’s requirement keeps changing. * In large system where it’s easy to find modules for incremental implementation. * When clients wants to start using the core features instead of waiting for the full software. * When there is a prediction that customer might add additional functionality in future. | * There is greater difficulty in managing requirements; * Greater efforts are needed to develop reusable units; * Greater efforts are needed for providing component specifications and additional material that help developers/consumers of the components. * The project duration is long. * When quality is more important than cost or schedule. * New version of existing product is needed. |

1. **Conclusion**

From the analysis done in the paper, the best SDLC model for a particular system can now be determined among Waterfall Model, Evolutionary Model, and Component-Based development so as to avoid project failure and loss. Evolutionary is suitable when customers want to start using the core module before delivery of the whole system(Techopedia Inc, 2021). Component-based development works on improving the quality of existing systems by building on previous models used to develop the system. (Techopedia Inc, 2021)

**References**

Adeagbo, M. A., Akinsola, J. E. T., Awoseyi, A. A., & Kasali, F. . (2017). *Project Implementation Decision Using Software Development Life Cycle Models : A Comparative Approach*. *2013*.

Akinsola, J. E. T., Ogunbanwo, A. S., & B, O. J. O. (2021). *Comparative Analysis of Software Development Life Cycle Models ( SDLC )*. *1*, 310–322. SDLC model · Software Development Life Cycle · Software engineering · Spiral model · V-Model · Waterfall model

Engineering, S. (2004). *Component-based software engineering*. 1–48. https://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/Ch19.pdf

Khan, A. I., Alam, M. M., Qayyum, N.-, & Khan, U. A. (2013). *An Improved Model for Component Based Software Development [11] \_ Download Scientific Diagram*. https://www.researchgate.net/publication/236108797\_Validation\_of\_Component\_Based\_Software\_Development\_Model\_using\_Formal\_B-Method

Muhammad, R., & Mahboob, M. (2016). *Enhancement in V- model for Component Based Development*. *7*(9), 25–30.

Techopedia Inc. (2021). *component-based-development-cbd*. https://www.techopedia.com/definition/31002/component-based-development-cbd