**LIRA UNIVERSITY**



**FACULTY: MANAGEMENT SCIENCE.**

**DEPARTMENT: COMPUTING AND INFORMATION**

**SCIENCES.**

**PROGRAMME: BACHELOR OF SCIENCE IN COMPUTER SCIENCE.**

**COURSE UNIT: SOFTWARE EVOLUTION.**

***LECTURER: MR. OMARA PATRICK.***

***COURSE CODE: LCS 3203***

***NAME: ARIONG BENARD.***

***REG. NO. 19/U/0011/LCS.***

***STUDENTS NO: 1901100637.***

**Individual coursework.**

**Submission Date: 14th/7/2022**

1. a) With examples discuss the Lehman’s laws of software evolution
2. Discuss the importance of reverse and forward engineering as a tool for re-engineering
3. a) Explain the various ways of evaluating legacy systems
4. What are the importance of legacy systems in software evolution?

**Answers**

1. **a) With examples discuss the Lehman’s laws of software evolution**

Software evolution is a very important activity where the software must have the ability to adapt according to the environment or user needs, to keep its satisfactory performance, given that if a system does not support changes, it will gradually lapse into uselessness

Meir Lehman started to formulate his laws of software evolution, after realizing the need for software systems to evolve. These laws stressed that a system needed to evolve due to its requirement to operate in or address a problem or activity in the real world, what Lehman called E-type Software. However, the laws were evaluated in the context of single systems.

* **Continuing change** — A software will become progressively less satisfying to its users over time, unless it is continually adapted to meet new needs.
* **Increasing complexity** — A software system will become progressively more complex over time, unless explicit work is done to reduce its complexity.
* **Self-regulation** — The process of software evolution is self-regulating, with close to normal distribution of the product and process artifacts that are produced.
* **Conservation** **of** **organizational** **stability** — The average effective global activity rate on an evolving software system does not change over time; that is, the amount of work that goes into each release is about the same.
* **Conservation** **of** **familiarity** — The amount of new content in each successive release of a software system tends to stay constant or decrease over time.
* **Continuing** **growth** — The amount of functionality in a software system will increase over time, in order to please its users.
* **Declining** **quality** — A software system will be perceived as declining in quality over time, unless its design is carefully maintained and adapted to new operational constraints.
* **Feedback** **System** — Successfully evolving a software system requires recognition that the development process is a multi-loop, multi-agent, multi-level feedback system; thus, for example, as a software system ages, it tends to become increasingly difficult to change due to the complexity of both the artifacts as well as the processes involved in effecting change. This law also implicitly recognizes the role of user feedback in providing impetus for future evolution.

***Example***

An empirical investigation where four of the Lehman’s Laws (LL) of Software Evolution were used in an SPL industrial project to understand how the SPL assets evolve over time. This project relates to an application in the medical domain developed in a medium-size company in Brazil. It contains 45 modules and a total of 70.652 bug requests in the tracking system, gathered along the past 10 years. two techniques were employed - the KPSS Test and linear regression analysis, to assess the relationship between LL and SPL assets. Finally, results showed that three laws were supported based on the data employed (continuous change, increasing complexity, and declining quality). The other law (continuing growth) was partly supported, depending on the SPL evaluated asset (common, variable or product-specific).

1. **Discuss the importance of reverse and forward engineering as a tool for re-engineering**

**Software Re-engineering** is a process of software development which is done to improve the maintainability of a software system. Re-engineering is the examination and alteration of a system to reconstitute it in a new form. This process encompasses a combination of sub-processes like reverse engineering, forward engineering, reconstructing among others.

**Reverse Engineering:**

Reverse Engineering is also known as backward engineering, is the process of forward engineering in reverse. In this, the information is collected from the given or existing application. It takes less time than forward engineering to develop an application. In reverse engineering, the application is broken to extract knowledge or its architecture.



The reverse engineering process includes taking apart worn-down products to examine how individual parts work, and then incorporating past inventions into new ones.

***Importance of Reverse Engineering***

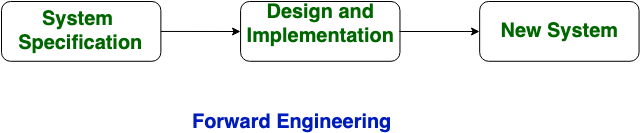
* ***Exploring existing designs and manoeuvres***  
  Reverse engineering allows us to see what already exists. This includes any parts, structures, or processes that could benefit communities in other ways. Examining current products leads to innovation and discovery, all thanks to reverse engineering.
* ***Reconstructing a product that is outdated***  
  A key part of redesigning an existing product is understanding the product itself. Reverse engineering provides the visual to work out outdated kinks in an older system. Quality is the most important aspect of this process.
* ***Discovering any product vulnerabilities***  
  Similar to the previous step, reverse engineering supports finding faults in the product. This is to ensure the safety and well-being of the product’s users. It is best for an issue to arise during the research phase rather than the distribution phase.
* ***Bringing less expensive & more efficient products to the market***  
  Reverse engineering’s main goal is to lead engineers on a path towards innovation and success. Succeeding includes lowering manufacturing costs and raising product effectiveness as much as possible.
* ***Creating a reliable CAD model for future reference***  
  Most reverse engineering processes include a full-working CAD file for future references. A CAD file is created so the part can be examined digitally if future issues arise. This form of technology has enhanced engineering productivity and product expression.
* ***Inspiring creative minds with old ideas***  
  Lastly, reverse engineering gives way for innovative design. During the process, an engineer might discover a system that could be useful for a completely different project. This shows how engineering connects projects with previous knowledge.

***Forward Engineering***

Forward engineering is the process of building from a high-level model or concept to build in complexities and lower-level details. Forward Engineering method is used to create or make an application with the help of the given requirements. Forward engineering is also known as Renovation and Reclamation. Forward engineering requires high proficiency skills. It takes more time to construct or develop an application. This type of engineering has different principles in various software and database processes.

. Forward Engineering applies of all the software engineering process which contains SDLC to recreate associate existing application. It is near to full fill new needs of the users into re-engineering.

Forward engineering tools are accustomed move from implementation styles and logic to the event of supply code. These tools basically follow the top-to down approach. System creator and visual Analyst is a forward engineering CASE tool.



***Importance of forward engineering***

Forward engineering is important in IT because it represents the 'normal’ development process. For example, building from a model into an implementation language. This will often result in loss of semantics, if models are more semantically detailed, or levels of abstraction.

1. **a) Explain the various ways of evaluating legacy systems**

A legacy system is an obsolete computer system, programming language, software application, process, or technology that is no longer can be maintained, replaced, or easily updated. It does not mean that the legacy system is unusable. Many organizations or companies still find these systems essential to their daily work. It depends upon you to either upgrade or replaces it. When evaluating your legacy tools, you need to logically identify issues with your existing solution that affecting your business.

**Legacy systems can be evaluated through the following way.**

Step 1 - Complete a Post-it Note Analysis

**Identify the top three issues and the top three benefits of your system. Write them down on sticky notes.**And keep in mind the following factors:

* **Stability and Reliability**– Does the system stay live consistently or does your team experience downtime too frequently?
* **Maintainability** – Is the system easy to maintain? Can maintenance be done internally? Is the company that built the software still thriving? Is it offering support as your partner, or are they just another vendor? and willing to help?
* **Compatibility** - Are you able to integrate new software to continually improve your system? Is it easy or difficult to connect with other systems?
* **Outdated Function** – Does the software still do everything you need it to do? Is your team creating and using “hidden” manual workarounds?

## Step 2 - Estimate the Severity

Legacy system drag has real business costs—both direct and indirect. You need to put numbers on both. Follow this assessment:

* If the current system is often experiencing downtime, what opportunities are missed during that downtime?
* If the system provider no longer supports the technology, are there other vendors that can?
* How much time is being wasted?
* How much money is associated with that lost time or lost opportunity?
* How deeply is poor user experience contributing to staffing or customer attrition rates?

Step 3 - Identify a Task Force

Put together a team of internal and external users and stakeholders who would benefit most from legacy system optimization—or have the most to lose from not taking action. Start by identifying user types, then choose a representative from each group. Involve customers and/or partners and vendors where needed—you can do that with a quick, simple email survey to assess ask what’s working, and what’s not.

Step 4 - Complete a Deeper Assessment (SWOT)

Complete a [SWOT analysis](https://www.wordstream.com/blog/ws/2017/12/20/swot-analysis). Each user type representative can work through a SWOT analysis for the parts of the system they use most. Below are a few questions from each category to aid in your system analysis.

* **Strengths** – What is currently working well in your system? What do you like about the workflow that makes tasks easier? Does the system process things quickly?
* **Weaknesses** – Are there times when you need to enter data into two or more different systems that don’t sync with one another? Does the current workflow leave too much room for human error by needing too many manual entries? Is there a lack of support from your current vendor?
* **Opportunities** – Can you process more orders? Can you improve the customer experience? Can you save money by consolidating or integrating existing workflows?
* **Threats** – Will you fall behind the competition? Does your system get in the way of exceptional customer service? Are employee onboarding and retention stressed by software or organizational shortcomings?

Step 5 - Compare Your Options to Make Informed Decisions

Now that you’ve outlined the issues, complied a team, and identified areas that require priority attention, you’re ready to weigh your options for action.

**What are the options?**

* Hire a consultant for a deeper analysis
* Make updates to the current system with your incumbent provider
* Implement integrated solutions to keep current functionality and improve productivity
* Start devising a plan to replace your system with a brand-new solution
  + Review both off-the-shelf and custom solutions

**Questions to consider when determining the right decision for your company…**

* Do you have or know a trustworthy development partner who understands the software?
* What is the cost difference between making updates and buying new? Consider integrations to mitigate costs and risks!
* Is the plan for updating going to tackle a symptom of an outdated system or the root cause?
* Do you need to hire a consultant or get a business analysis to get a deeper assessment?
  1. **What is the importance of legacy systems in software evolution?**

Before you start to replace or to upgrade your legacy system, take a step back, and understand the positive impact that your legacy system can have. Knowing the contribution legacy system makes so that when you want to replace the system, the new solution must deliver the same or more enhanced solution as the old one.

* **Work well enough.** Many legacy systems are doing their job just fine. The proverb “if it isn’t broke, don’t fix it” applied even when you need to run your legacy application on a mainframe instead of cloud computing.
* **Converting existing legacy data to a new system isn’t easy.** The way data is stored and retrieved is changing. An older system may have a different algorithm, and some are no longer common. To convert all of the data into an entirely different form to be stored in a modern computing system are quite challenging on the internal IT team.
* **Recertifying and revalidating require considerable effort and cost.** In some industries, software that can impact safety must be validated and certified for compliance. This software is critical to the company or customer. Recertifying and validating the new system may take a long time and a high cost.
* **Switching the tool is too disruptive to the organization.** The downtime, while transitioning to new software, is considered an escalating cost to pay because of a decrease in productivity when the new software is deployed and learned.
* **The cost of replacing is higher than the benefits of a new system.** There is monetary value in replacing something old with something new. The reduction in cost and increase in business should be significant than the total cost of switching.
* **Legacy systems ensure continuity in business operations.** Modernizing software or switching to entirely new technology is a delicate process that could put business operations on hold. Alternatively, maintaining the status quo (legacy systems) lets business operations run smoothly with no hitches or pauses.
* **Legacy systems are familiar.** Having a legacy system provides a unique feeling of familiarity and comfort. In actuality, this close attachment is among the most common reasons for retaining legacy systems.
* **Legacy systems are efficient for specific tasks.** Efficiency is another legacy software advantage. Some legacy systems, despite being old and outdated, still work perfectly fine. And most importantly, these systems are typically designed for maximum capacity and remain reliable and durable in most cases.

**REFERENCES**

<https://plg.uwaterloo.ca/~migod/papers/2013/lehmanPaper.pdf>

<https://www.cedarvilleeng.com/news-and-insights/blog/6-benefits-of-reverse-engineering/>

<https://link.springer.com/chapter/10.1007/978-3-319-14130-5_4?noAccess=true>

<https://astrixinc.com/evaluating-legacy-lims-software/>

<https://www.sciencedirect.com/science/article/abs/pii/S0950584908000694>