



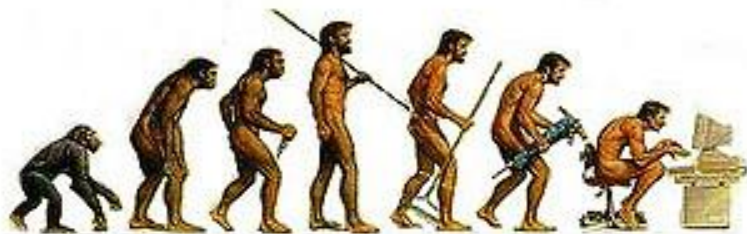
# Software Evolution

Prepared by:  
Chrevic Josef P. Dangan

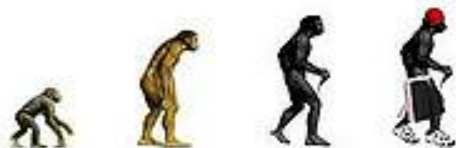


EVOLVE

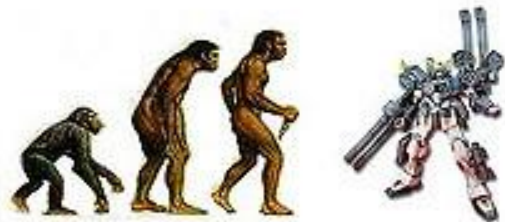




***EUROPE***

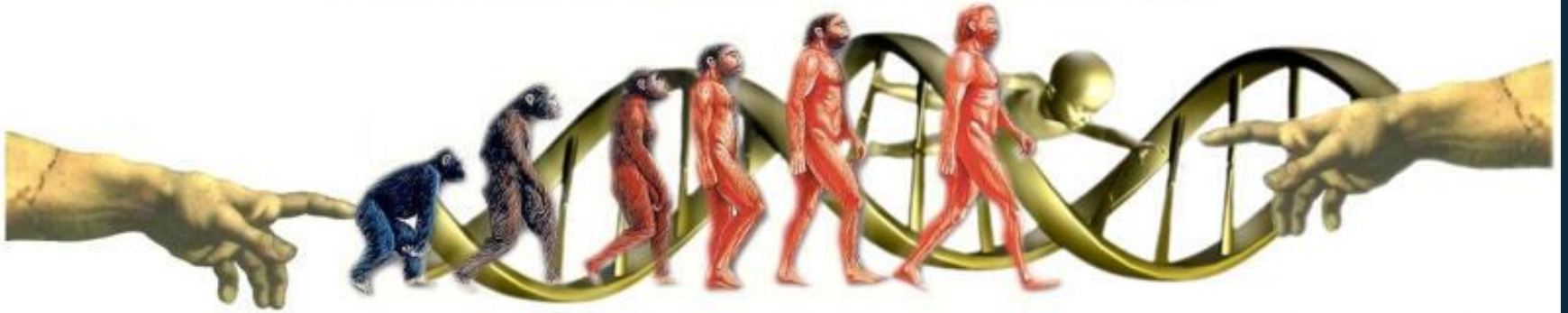


***AMERICA***



***JAPAN***

**WHETHER EVOLVED OR ENDOWED**



**WHAT MOST DISTINGUISHES HUMAN BEINGS FROM ALL OTHER LIFE FORMS IS**

**OUR CAPACITY FOR LEARNING**



# Today's Topics

- ◇ Evolution processes
- ◇ Program evolution dynamics
- ◇ Software maintenance
- ◇ Legacy system management





# Software change

- ◇ Software change is inevitable
  - New requirements emerge when the software is used;
  - The business environment changes;
  - Errors must be repaired;





# Software change

- ◇ Software change is inevitable
  - New computers and equipment is added to the system;
  - The performance or reliability of the system may have to be improved.





# Software change

- ◇ The key problem for all organizations is implementing and managing change to their existing software systems.







# Importance of evolution

- ◇ Organizations have huge investments in their software systems - they are critical business assets.
- ◇ To maintain the value of these assets to the business, they must be changed and updated.



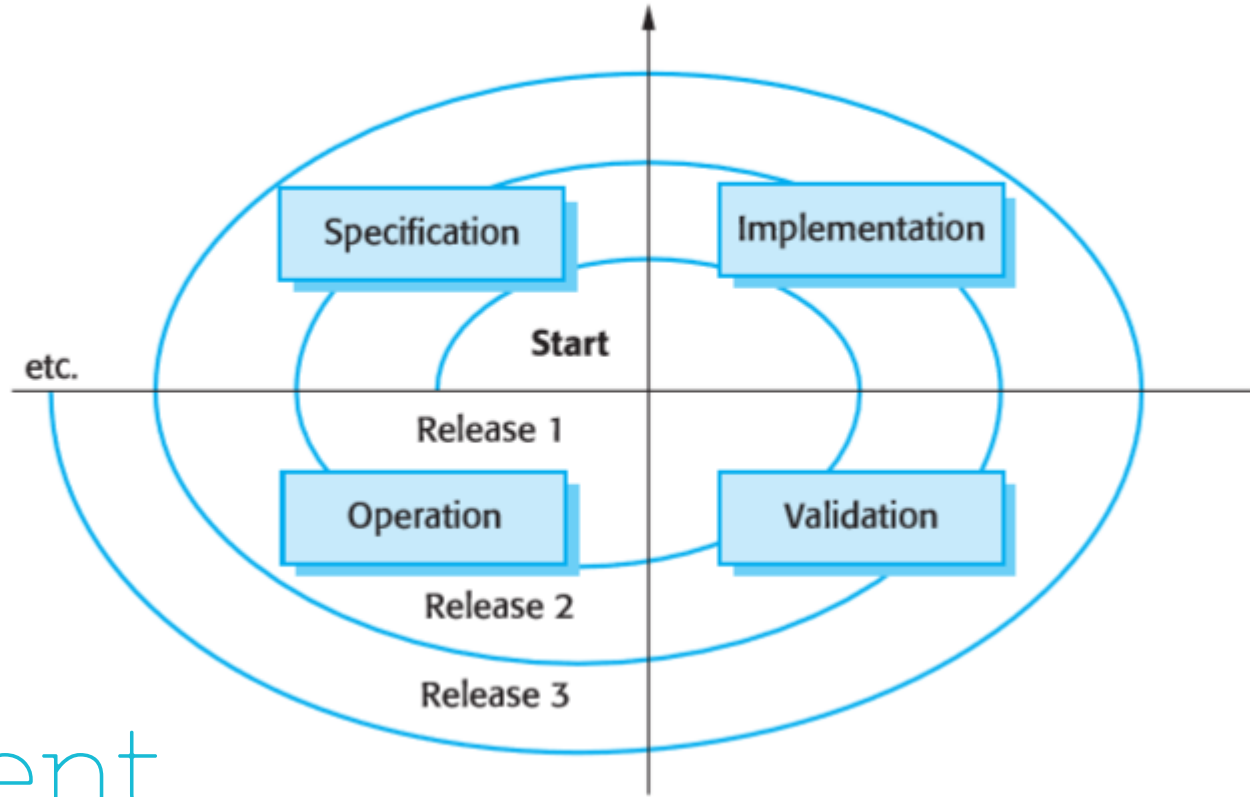


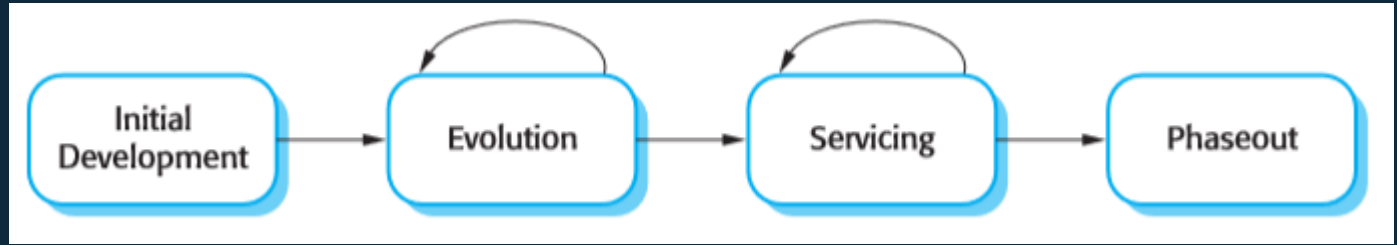
# Importance of evolution

- ◇ The majority of the software budget in large companies is devoted to changing and evolving existing software rather than developing new software.



# spiral model of development and evolution





Evolution  
and  
servicing



# Evolution and servicing

## ◇ Evolution

- The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system.





# Evolution and Servicing

## ◇ Servicing

- The software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software's environment. No new functionality is added.





# Evolution and Servicing

## ◇ Phase-out

- The software may still be used but no further changes are made to it.





# Evolution Processes

## ◇ Software evolution processes depend on

- The type of software being maintained
- The development processes used
- The skills and experience of the people involved







# Evolution Processes

- ◇ **Proposals for change are the driver for system evolution**
  - Should be linked with components that are affected by the change, thus allowing the cost and impact of the change to be estimated.



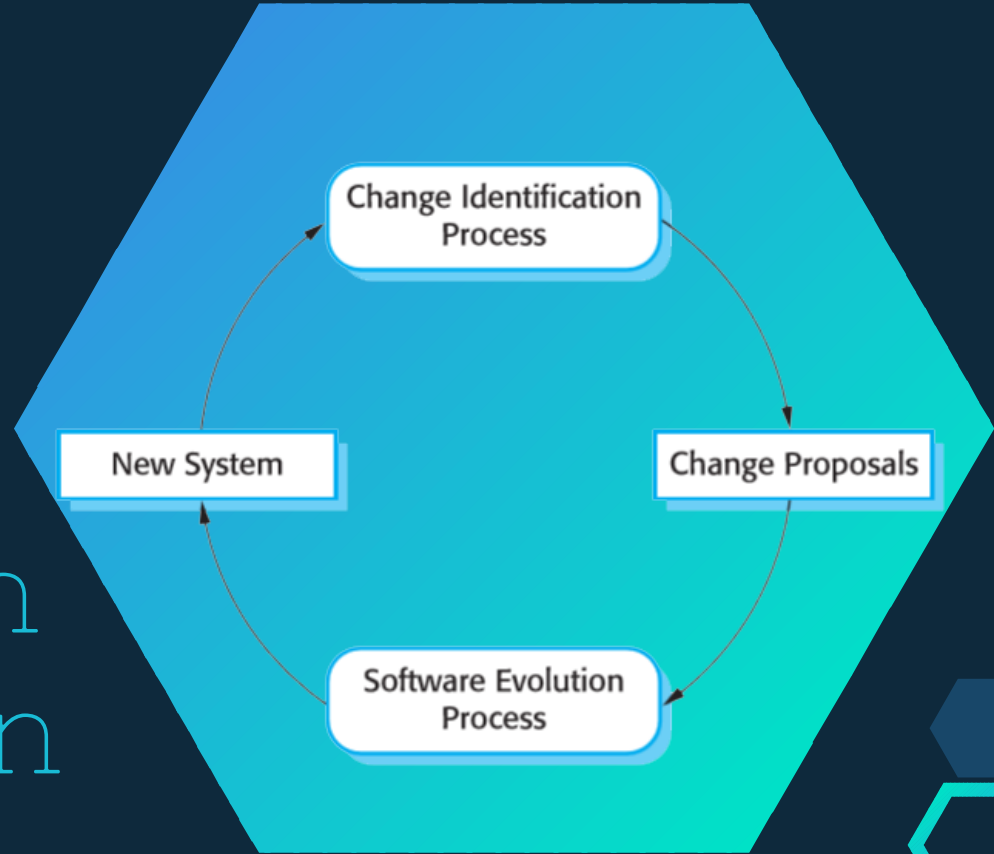


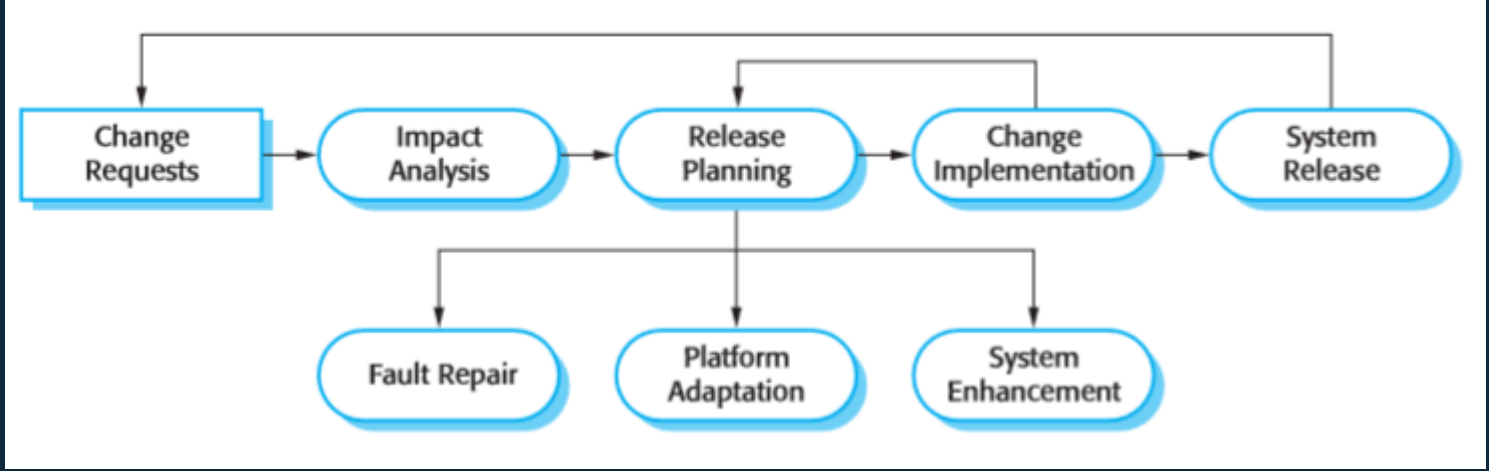
# Evolution Processes

- ◇ Change identification and evolution continues throughout the system lifetime.



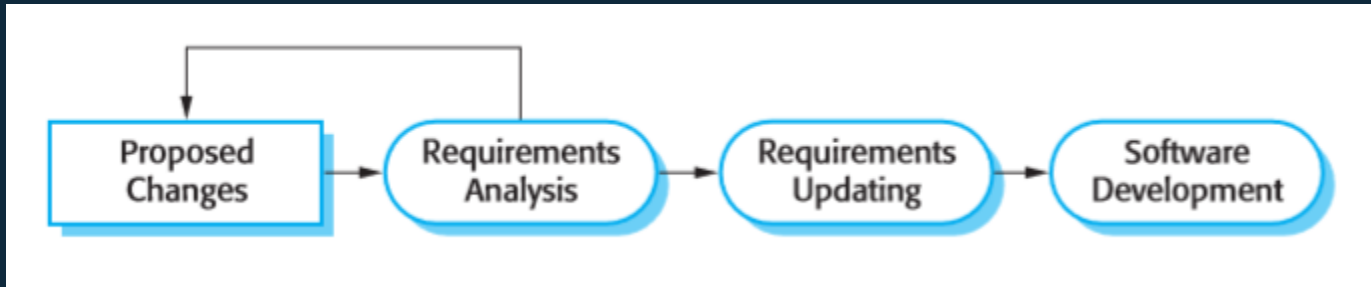
# Change identification and evolution processes





# Software Evolution Process





# Change Implementation



# Change Implementation

- ◇ Iteration of the development process where the revisions to the system are designed, implemented and tested.





# Change Implementation

- ◇ A critical difference is that the first stage of change implementation may involve program understanding, especially if the original system developers are not responsible for the change implementation.





# Change Implementation

- ◇ During the program understanding phase, you have to understand how the program is structured, how it delivers functionality and how the proposed change might affect the program.







# Urgent change requests

- ◇ Urgent changes may have to be implemented without going through all stages of the software engineering process
  - If a serious system fault has to be repaired to allow normal operation to continue





# Urgent change requests

- ◇ Urgent changes may have to be implemented without going through all stages of the software engineering process
  - If changes to the system's environment (e.g. an OS upgrade) have unexpected effects

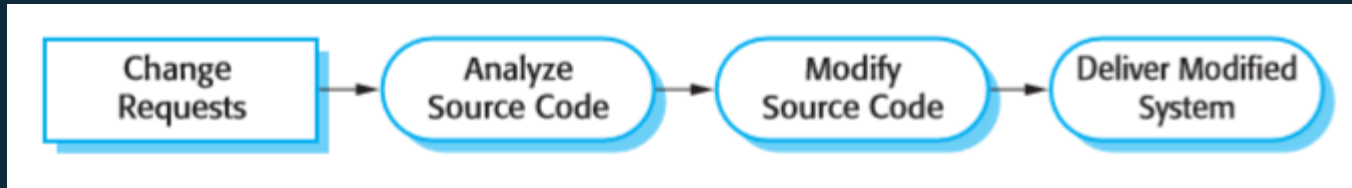




# Urgent change requests

- ◇ Urgent changes may have to be implemented without going through all stages of the software engineering process
  - If there are business changes that require a very rapid response (e.g. the release of a competing product)





The emergency  
repair process



# Agile methods and evolution

- ◇ **Agile methods are based on incremental development so the transition from development to evolution is a seamless one.**
  - Evolution is simply a continuation of the development process based on frequent system releases.





# Agile methods and evolution

- ◇ Automated regression testing is particularly valuable when changes are made to a system.






# Agile methods and evolution

- ◇ **Changes may be expressed as additional user stories.**






# Handover problems

- ◇ Where the development team have used an agile approach but the evolution team is unfamiliar with agile methods and prefer a plan-based approach.
  - The evolution team may expect detailed documentation to support evolution and this is not produced in agile processes.







# Handover problems

- ◇ **Where a plan-based approach has been used for development but the evolution team prefer to use agile methods.**
  - The evolution team may have to start from scratch developing automated tests and the code in the system may not have been refactored and simplified as is expected in agile development.





# Program evolution dynamics

- ◇ **Program evolution dynamics** is the study of the processes of system change.
  - After several major empirical studies, Lehman and Belady proposed that there were a number of ‘laws’ which applied to all systems as they evolved.





# Program evolution dynamics

- ◇ There are sensible observations rather than laws. They are applicable to large systems developed by large organizations.
  - It is not clear if these are applicable to other types of software system.





# Change is Inevitable

- ◇ The system requirements are likely to change while the system is being developed because the environment is changing. Therefore a delivered system won't meet its requirements!





# Change is Inevitable

- ◇ Systems are tightly coupled with their environment. When a system is installed in an environment it changes that environment and therefore changes the system requirements.





# Change is Inevitable

- ◇ Systems **MUST** be changed if they are to remain useful in an environment.





# Lehman's Laws

## ◇ Continuing change

- A program that is used in a real-world environment must necessarily change, or else become progressively less useful in that environment.





# Lehman's Laws

## ◇ Increasing complexity

- As an evolving program changes, its structure tends to become more complex.
- Extra resources must be devoted to preserving and simplifying the structure.







# Lehman's Laws

## ◇ Large program evolution

- Program evolution is a self-regulating process.
- System attributes such as size, time between releases, and the number of reported errors is approximately invariant for each system release.





# Lehman's Laws

## ◇ Organizational stability

- Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.





# Lehman's Laws

## ◇ Conservation of familiarity

- Over the lifetime of a system, the incremental change in each release is approximately constant.





# Lehman's Laws

## ◇ Continuing growth

- The functionality offered by systems has to continually increase to maintain user satisfaction.





# Lehman's Laws

## ◇ Declining quality

- The quality of systems will decline unless they are modified to reflect changes in their operational environment.





# Lehman's Laws

## ◇ Feedback system

- Evolution processes incorporate multi-agent, multi-loop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.





# Applicability of Lehman's laws

- ◇ **Lehman's laws seem to be generally applicable to large, tailored systems developed by large organizations.**
- Confirmed in early 2000's by work by Lehman on the FEAST project.





# Applicability of Lehman's laws

## ◇ It is not clear how they should be modified for

- Shrink-wrapped software products
- Systems that incorporate a significant number of COTS components
- Small organizations
- Medium sized systems.







# Key Points

- ◇ Software development and evolution can be thought of as an integrated, iterative process that can be represented using a spiral model.
- ◇ For custom systems, the costs of software maintenance usually exceed the software development costs.





# Key Points

- ◇ The process of software evolution is driven by requests for changes and includes change impact analysis, release planning and change implementation.



A cluster of hexagons in the top-left corner, featuring a large cyan hexagon, a smaller cyan hexagon above it, and several dark blue and light blue hexagons around it.

# Key Points

- ◇ Lehman's laws, such as the notion that change is continuous, describe a number of insights derived from long-term studies of system evolution.





Any questions?



# References



*Software Engineering*, 9th Edition,  
I.Sommerville



# Last Reaction Paper

Learning from Stack  
Overflow

[https://www.youtube.com/  
watch?v=NWHfY\\_1vKIQ1](https://www.youtube.com/watch?v=NWHfY_1vKIQ1)



# Next Meeting

## Foundations of Security

