

Software Quality Engineering

Week 3

Engr. Muhammad Umer Haroon

FOLLOW THE RULES

- ▶ Use of Mobile (no)
- ▶ Late coming (no)
- ▶ Short Attendance (no)
- ▶ Plagiarism (no)
- ▶ Cheat (no)
- ▶ Disrespect (no)

Quality attributes



SOFTWARE QUALITY MODELS

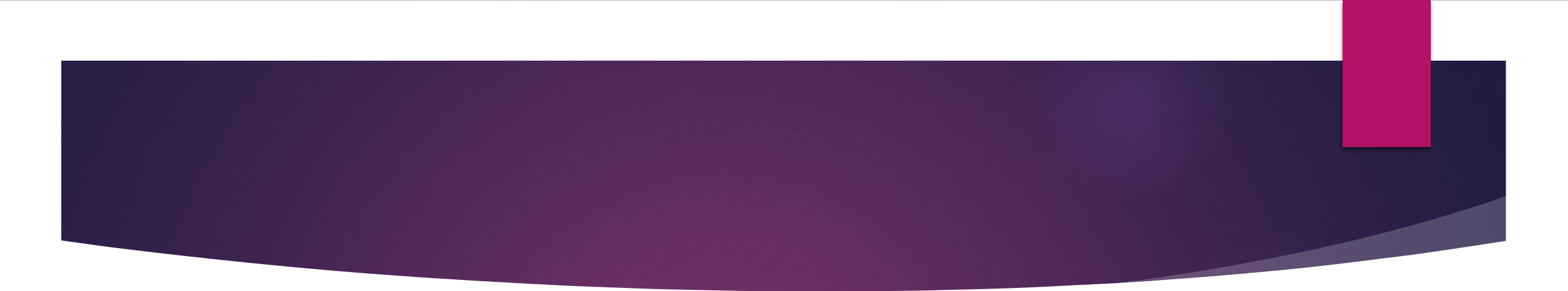
Quality is the excellence of the product or service.

- ▶ From a user's point of view, quality is 'fitness for purpose'.
- ▶ From the manufacturing point of view, the quality of a product is the conformance to specification.

Hierarchical models

McCall divided software quality attributes into 3 groups.

- ▶ Each group represents the quality with respect to one aspect of the software system while the attributes in the group contribute to that aspect.
- ▶ Each quality attribute is defined by a question so that the quality of the software system can be assessed by answering the question.

- 
- ▶ The McCall model is a framework for evaluating software quality, and it provides a way to measure the quality of a software system in terms of its ability to meet the needs of its users.
 - ▶ Jim McCall produced the McCall software quality model for the US Air Force in 1977. This is used to maintain harmony between the users and the developers. Successful software is developed that fulfills the user needs in consideration with the developer's point of view.

Software Quality Attributes

Maintainability - *Can I fix it?*

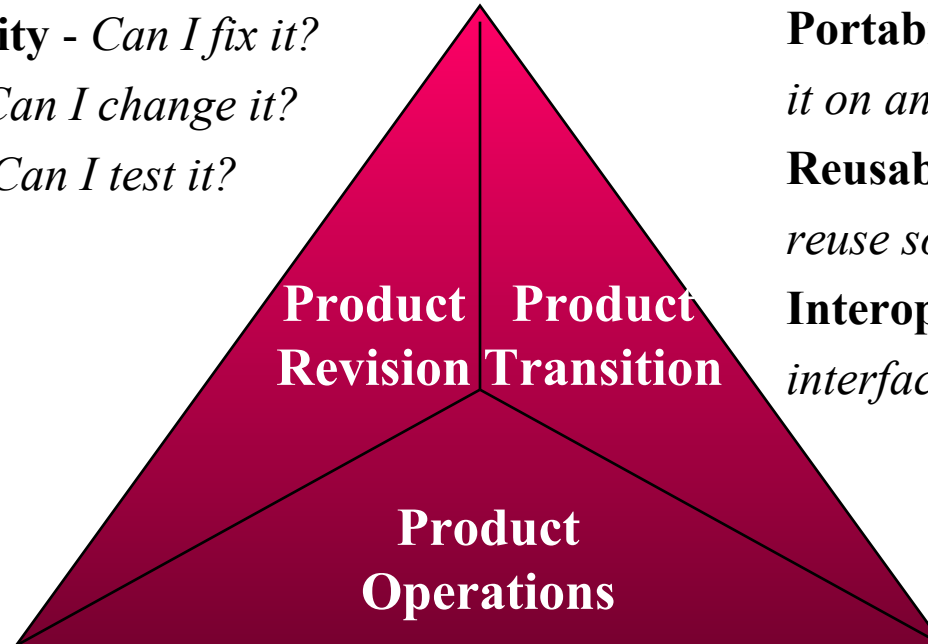
Flexibility - *Can I change it?*

Testability - *Can I test it?*

Portability - *Will I be able to use it on another machine?*

Reusability - *Will I be able to reuse some of the software?*

Interoperability - *Will I be able to interface it with another machine?*



Correctness - *Does it do what I want?*

Reliability - *Does it do it accurately all the time?*

Efficiency - *Will it run on my machine as well as it can?*

Integrity - *Is it secure?*

Usability - *Can I easily use it?*

Product Revision

- ▶ It encompasses the revision perspective identifiers quality factors that changes or enhances the ability to change the software product in the future according to the needs and requirements of the user.
- **Maintainability**- If there are defects in the software that are found in the later stage, this feature allows finding and fixing the defects.
- **Flexibility**- The ability to make changes in the software product according to the business demands.
- **Testability**- This enables the software product to validate the requirements.

Product Transition

- ▶ Transition perspective enables the software to adapt itself in new environments. The identification of the quality factor which enables the ability of adaption of the software in the new environment is known as product transition.
- ▶ Let us take an example, our creator created the software named human. The creator incorporated ability to this software that enhances it to adapt in the new environment.
 - **Portability**- This is the ability to transfer a software from one environment to another environment.
 - **Re-usability**- The software components can be used in different contexts.
 - **Interoperability**- The ease or the comfort zone in which all the components of the software works together.

Product Operations

- ▶ The software can run successfully in the market if it according to the specifications of the user and also it should run smoothly without any defects. The product operation perspective influences the extent to which the software fulfills its specifications-
- **Correctness**- The functionality should match the specification.
- **Reliability**- The extent to which the system fails.
- **Efficiency**- It enhances the usage of system resource.
- **Usability**- The software should be easy to use. Difficult software is tedious to work upon and difficulty irks the user.

Relational models

Perry's relational model contains three types of relationship between the quality attributes.

- ▶ The direct relationship
- ▶ The inverse relationship
- ▶ The neutral relationship

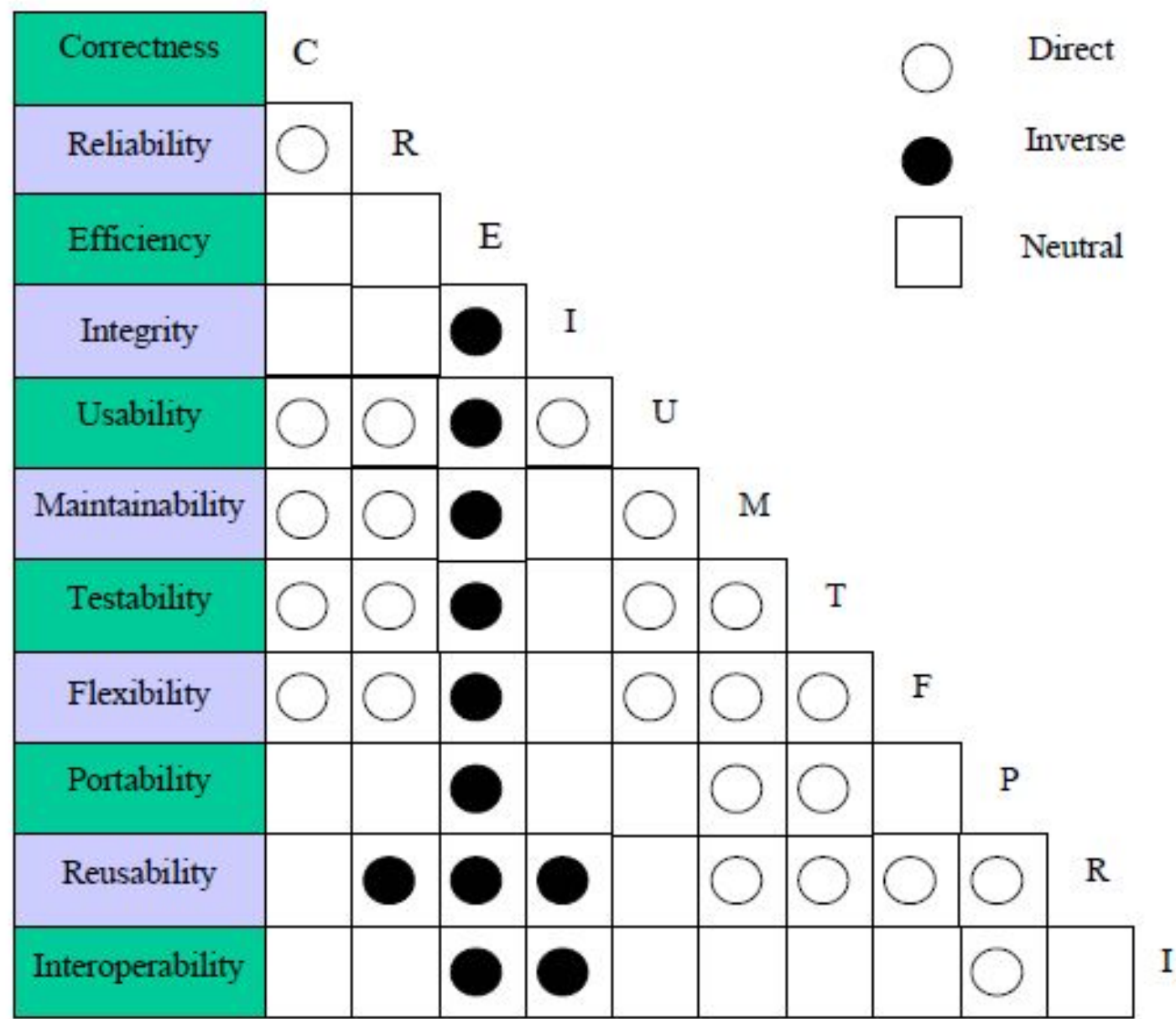
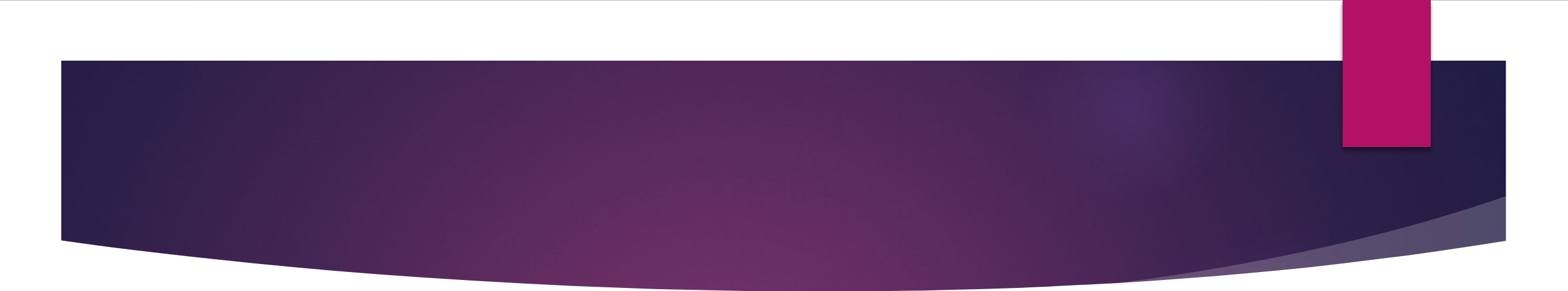


Figure 2.2 Perry's relational model of software quality

- 
- ▶ The direct relationship between two quality attributes means that if a software system is good at one attribute it should also be good at the other attribute.
 - ▶ The inverse relationship means that a software system that is good at one attribute will not be good at the other attribute.
 - ▶ The neutral relationship means that the two attributes are normally independent of each other.

Relationship b/w quality attributes

- ▶ *Integrity vs. efficiency (inverse):*

The control of data access will need additional code, leading to a longer runtime and more storage requirement.

- ▶ *Usability vs. efficiency (inverse):*

Improvement of HCI will need more code and data, hence the system will be less efficient.

Relationship b/w quality attributes

- ▶ *Maintainability and testability vs. efficiency (inverse):*
Compact and optimized code is not easy to maintain and test.
- ▶ *Flexibility, reusability vs. integrity (inverse):*
Flexible data structures required for flexible and reusable software increase the data security problem.
- ▶ *Reusability vs. maintainability (direct):*

Relationship b/w quality attributes

- ▶ *Portability vs. reusability (direct):*

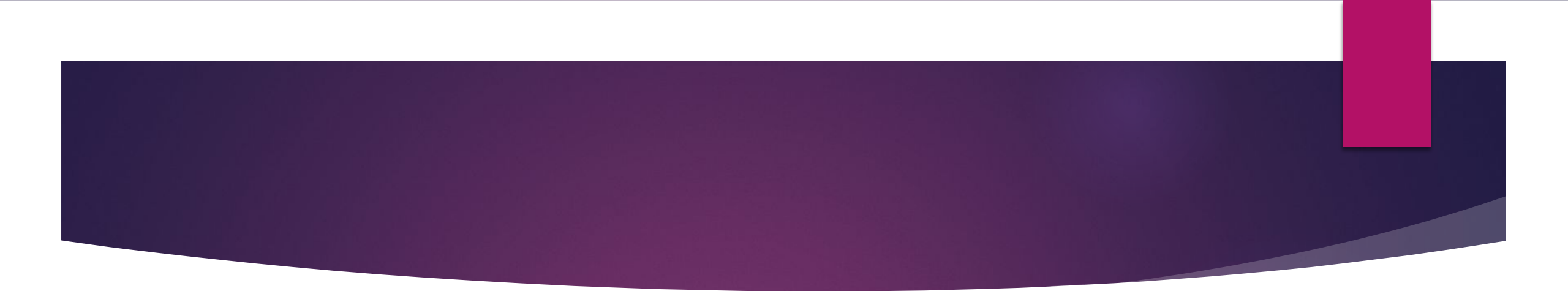
Portable code is likely to be easily used in other environments. The code is likely well-structured and easier to be reused.

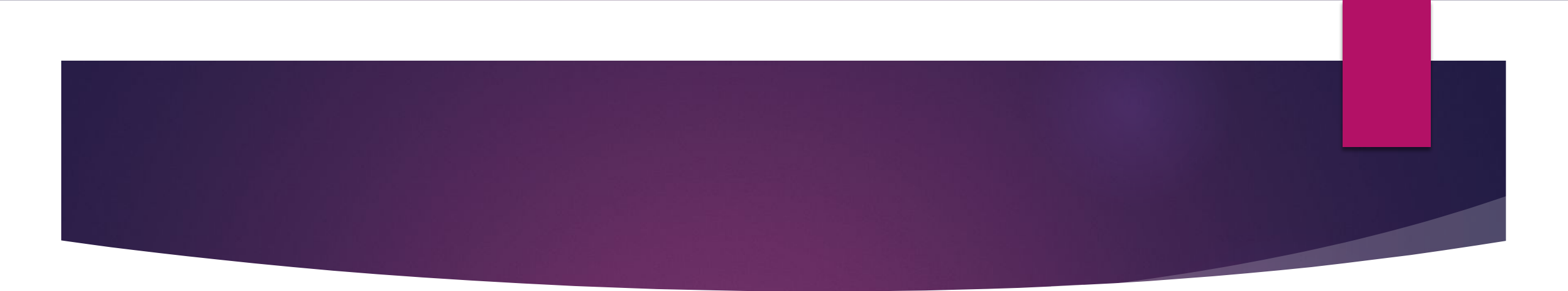
- ▶ *Correctness vs. efficiency (neutral):*

The correctness of code has no relation with its efficiency. Correct code may be efficient or inefficient in operation.

FURPS

- ▶ FURPS is a quality model for software development that provides a structured and comprehensive approach for evaluating and assessing software quality. It stands for Functionality, Usability, Reliability, Performance, Supportability.

- 
- ▶ FURPS is important in software quality because it helps organizations and developers to identify and prioritize the key quality factors that are critical for success in a given software project.
 - ▶ By using FURPS, software development teams can ensure that their software meets the functional, technical, and user-centered requirements of the stakeholders, and that it is delivered with high levels of reliability, performance, and supportability.

- 
- ▶ Additionally, FURPS helps to ensure that software quality is considered throughout the entire software development life cycle, from the initial planning and design stages, to the final delivery and ongoing maintenance phases.
 - ▶ This helps to minimize the risk of defects, improve the user experience, and increase the overall success and adoption of the software.



Usability Reliability

Functionality

- ▶ This element of FURPS covers the functional requirements of the software and includes the features, capabilities, and services that the software must provide.

Usability

- ▶ Usability refers to the ease of use and user-centered design of the software. This includes factors such as user interface design, navigation, and user assistance.

Reliability

- ▶ Reliability refers to the dependability and stability of the software. This includes aspects such as software robustness, fault tolerance, and error handling.

Performance

- ▶ Performance refers to the speed and efficiency of the software. This includes factors such as response time, processing speed, and resource utilization.

Supportability

- ▶ Supportability refers to the ability to maintain, diagnose, and troubleshoot the software. This includes factors such as maintainability, configurability.

Example if FURP is used for website

- ▶ **Functionality:** Ensure that the website meets the necessary functional requirements, such as product search, product display, and purchase capabilities.
- ▶ **Usability:** Ensure that the website is user-friendly and easy to navigate.
- ▶ **Reliability:** Ensure that the website is stable and operates consistently and reliably.
- ▶ **Performance:** Ensure that the website loads quickly and can handle high levels of traffic and transactions.
- ▶ **Supportability:** Ensure that the website is easy to maintain and update and that technical support is readily available to users.

ISO 9126

- ▶ ISO 9126 is an international standard for software product quality. It provides a systematic and comprehensive model for evaluating software product quality. The standard consists of six quality characteristics: functionality, reliability, usability, efficiency, maintainability, and portability. Each quality characteristic is further divided into several sub-characteristics, which are used to evaluate the software product.

ISO/IEC 25000:2018

- ▶ The ISO 9126 standard has been widely used in the software industry since its release in 1991. However, it has been updated and replaced by ISO/IEC 25000:2018, which is the latest international standard for software product quality evaluation.
- ▶ The new standard provides a more comprehensive and up-to-date approach to evaluating software product quality, taking into account the changes and advancements in the software industry over the years. While ISO 9126 is still widely used and recognized, it is generally recommended to use the updated ISO/IEC 25000 standard for software product quality evaluation.

Difference b/w them

- ▶ ISO 25000:2014 (reviewed in 2020) improved from the previous standard in the context coverage, freedom from risk and satisfaction and criteria, which were overlooked in ISO 9126.
- ▶ Security in the ISO 25000 has more relevance, and it is an independent factor.

**Software Product
Quality**
ISO 25010

**Functional
Suitability**

Appropriateness
Accuracy
Compliance

Reliability

Availability
Fault tolerance
Recoverability
Compliance

**Performance
efficiency**

Time-
behaviour
Resource-
utilisation
Compliance

**Operability
(Useability)**

Appropriateness-
recogniseability
Learnability
Ease-of-use
Helpfulness
Attractiveness
Technical-
accessibility
Compliance

Security

Confidentiality
Integrity
Non-repudiation
Accountability
Authenticity
Compliance

Compatibility

Replace-
ability
Coexistence
Inter-
operability
Compliance

**Maintain-
ability**

Modularity
Reusability
Analyzability
Changeability
Modification
stability
Testability
Compliance

**Transfer-
ability**

Portability
Adaptability
Installability
Compliance

Reading task

- ▶ <https://www.ukessays.com/essays/computer-science/review-comparison-software-quality-6861.php>

Software Design & Quality

SOFTWARE QUALITY MUST
BE ADDRESSED DURING THE
WHOLE PROCESS OF
SOFTWARE DEVELOPMENT.



However, design is of particular importance in developing quality software for two reasons.

- ▶ First, design is the first stage in software system creation in which quality requirements can begin to be addressed. Errors made at this stage can be costly, even impossible, to be rectified.
- ▶ Second, design decisions have significant effects on the quality of the final product

The input and start point of designs

- ▶ ‘necessity is the mother of invention’.
- ▶ Without users’ requirements, there will be no software design.

The outcome and results of designs

- ▶ Design representation serves as the basis to conceptualise and compare various design decisions.
- ▶ The true output of design is more than just a plan or symbolic representation. MacLean et al.[*] pointed out that, the final output of a design also include what they call ‘design space’, which is a body of knowledge about the artefact, its environment, its intended use, and the decisions that went into creating the design.

*MacLean, A. , Bellotti, V. and Young, R., *What rationale is there in design?*, in *Human-Computer Interaction – INTERACT’90*, 1990, pp207–212.

Software design tasks

Software design tasks can be divided into several interrelated subtasks such as

- ▶ architectural design,
- ▶ interface design
- ▶ and detail design including algorithm and data structure design.

Architecture design

- ▶ Architecture design determines how a software system is decomposed into components and how these components are interconnected.

Interface design

- ▶ Interface design determines how the software interacts with its environment, such as human computer interactions (HCI).

Detail design

- ▶ Detail design determines the details of the implementation, such as the algorithms and data structures for implementing each functionality and component, and the programming language for coding.

THE EFFECT OF DESIGN ON SOFTWARE QUALITY

Efficiency

- ▶ Efficiency refers to the responsiveness of the system, i.e. the time required to respond to stimuli (events), or the number of events processed in some interval of time.

1. Efficiencytime to process/communicate

- ▶ The **time to process a sequence** of events can be divided into **three** parts.
- ▶ First, time is needed to communicate between different software components that collaborate to process an event. The structure of a software system determines how much time is required by the communication between components in terms of the amount of communications required and the means of communications and interactions between components.
- ▶ The amount of communication depends on how functions of a software system are grouped into components.
- ▶ A bad design would result in a large amount of communication between components; while a good design can keep communications within components hence avoid unnecessary communications between components.

Efficiencytime to process/communicate

example

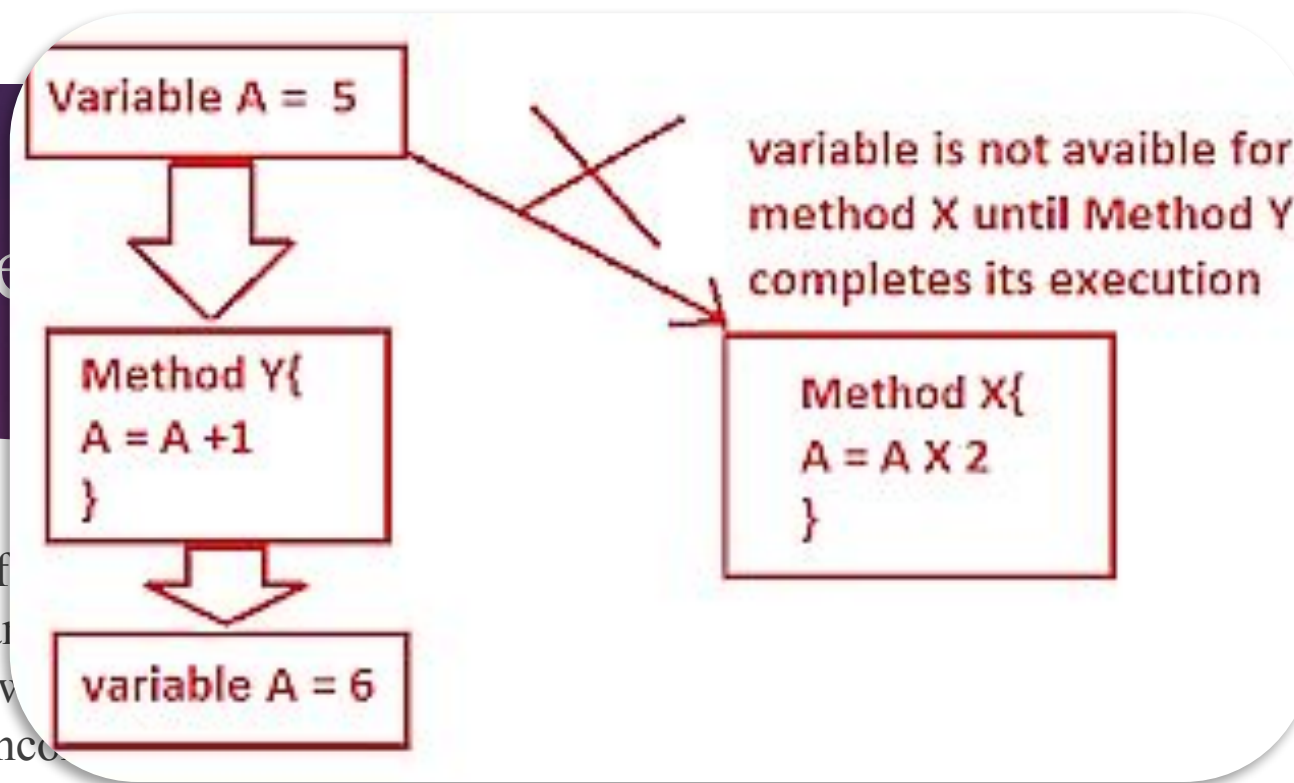
- ▶ The means of communication depends on the nature of the components involved.
- ▶ For example, if a component is a process, the communication must go through messages between processes.
- ▶ If the component is a process, the communication must go through messages between processes.
- ▶ **For instance, in a web browsing scenario, the client process (e.g. a web browser) sends a request message to the server process (e.g. a web server) to retrieve a webpage. The server process processes the request, retrieves the requested webpage, and sends a response message back to the client process with the webpage data. The client process then renders the webpage for the user to view.**
- ▶ The time required for a communication between components **also depends on how components are distributed on the network.** If the components reside on different computing elements, such as on different computers in a distributed system, the communication between these components has to go through a communication network, which takes much longer than communications between two components that reside on the same computer.
- ▶ However, even if the components are on the same computer, the amount of time required for interaction by subroutine invocations, message passing between processes, or other communication mechanisms still takes much longer than straight line codes and shared memory. **In general, communication tends to be a performance driver, which makes efficiency largely a function of architecture.**

2.Efficiency computation times

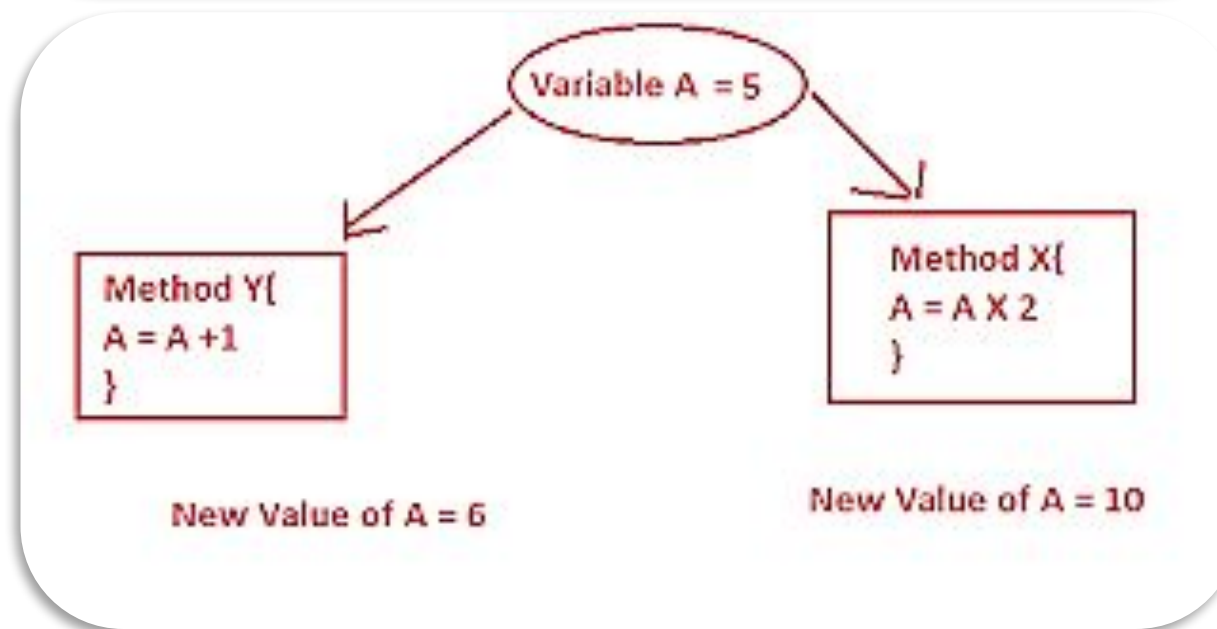
- ▶ Second, the computation times that components executed may have overlaps or gaps between them, i.e. the executions can be in parallel and time is required to synchronise their executions.
- ▶ Architectural design also determines how much parallel execution can be achieved and how long parallel processes have to spend on synchronisation and mutual exclusion.

Simplify the

- Synchronization refers to making threads work together correctly and in a predictable manner. Without synchronization, as race conditions, which can lead to incorrect results.



at they work
ent problems such
d order of events,



3.Efficiency computation times

- ▶ Third, the times are needed for each component to complete its computation. Once given an architectural design, the performance depends on the choice of algorithms and data structures to implement selected functionality assigned to the components.
- ▶ It also depends on how the algorithms are coded, e.g. in a particular programming language. The selection of algorithms, data structures and programming language is an issue of detailed design, while the coding is an implementation issue.

4. Efficiency

- ▶ The efficiency of the G₁ effect data set

efficiency of
though such
algorithm and

Correctness and reliability

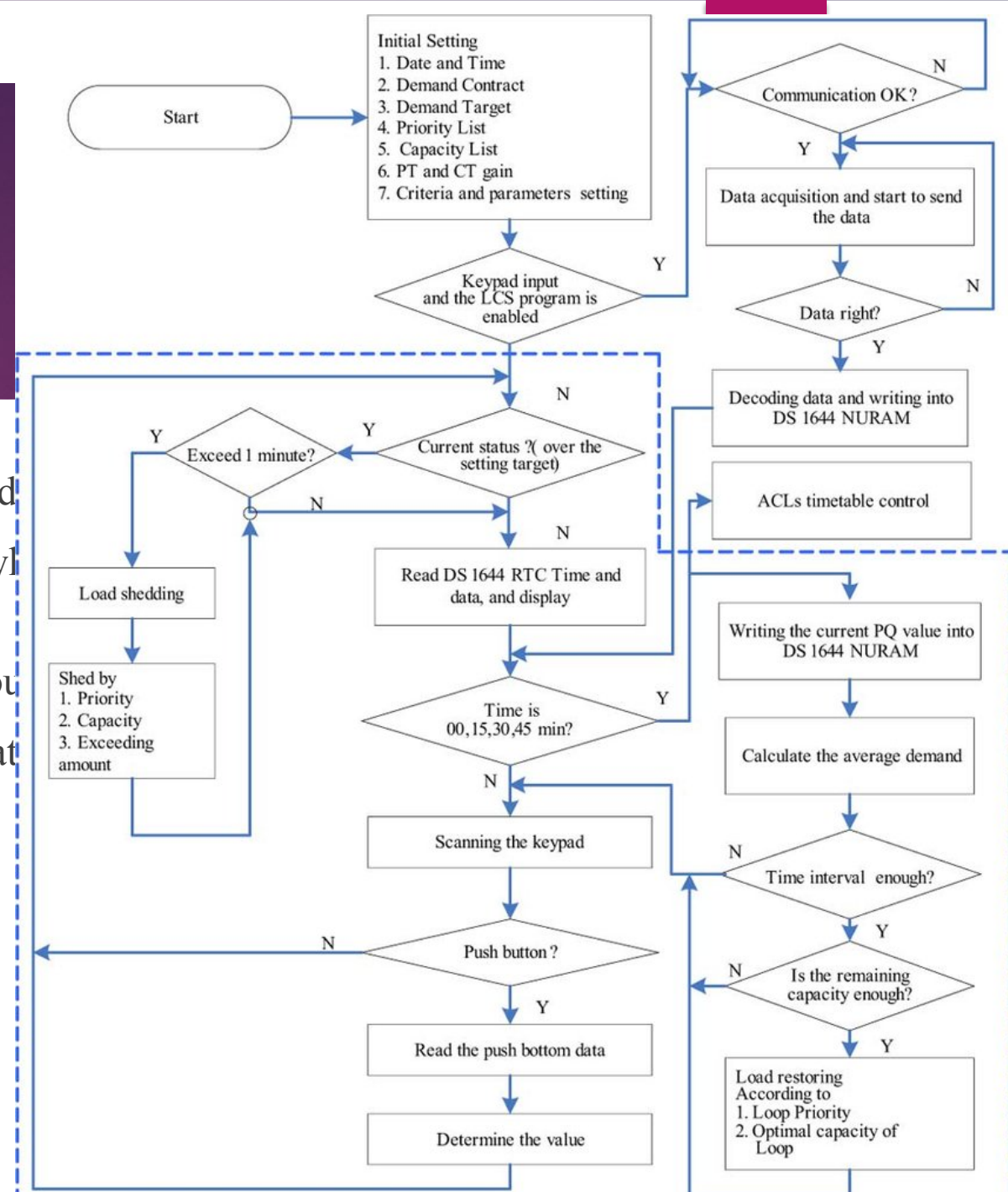
- ▶ Correctness is the property that software implements the specified users' requirements. It is impossible that a design at any level that does not correctly implement the specified requirements would lead to a correct implementation.
- ▶ Reliability can be defined as the probability that a system performs user required functionality correctly at a specified environment in a given period of time.

Correctness and reliability

- ▶ If architecture design has flaws , they will be carried to next phases in as errors in algos, bugs in code and eventually effecting reliability.
- ▶ Good HCI designs can prevent invalid input and misinterpretation of output; hence, they can significantly reduce the probability of system failures caused by human operation errors.

Correctness and reliability ..

- ▶ Detailed design such as algorithm and data structure d
- ▶ The **choice of programming language** determines wh naturally and easily coded.
- ▶ The choice of **algorithm** and **data structure** etc. shou
- ▶ Such factors have great impact on the probability that



Portability

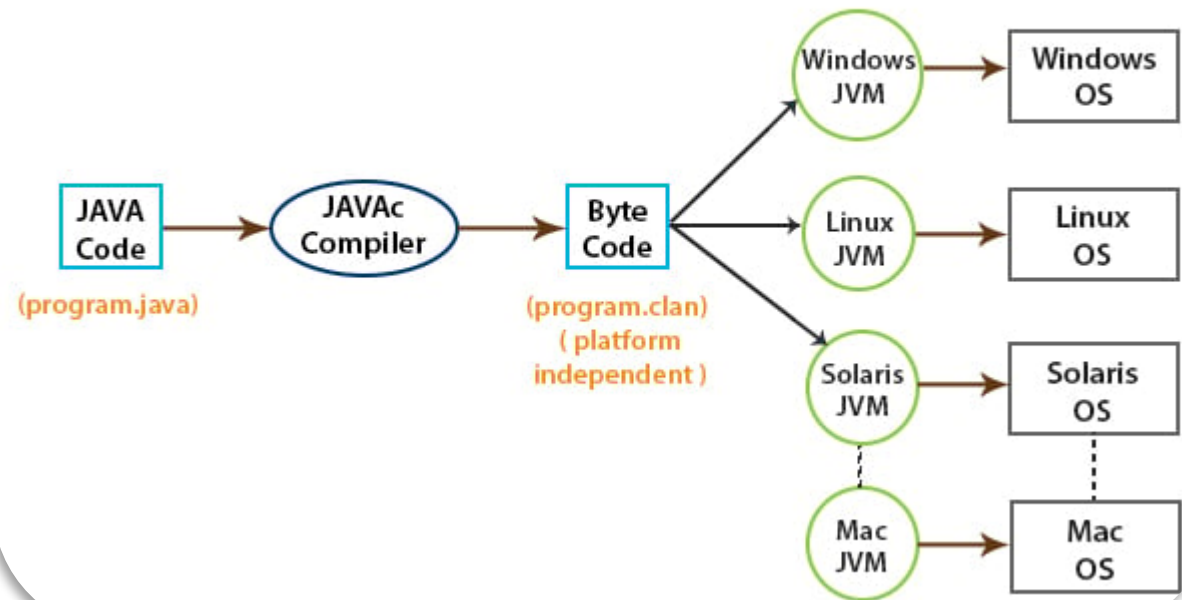
- ▶ Portability is the property of a software system that can be easily transported from one hardware/software platform to another, for example, from a PC/Windows environment to a Sun/Unix environment or a Macintosh environment.

Portability....

- ▶ Moving from one environment to another depends on the facilities provided by the environment
- ▶ A well structured architectural design of components so that the change on the components with new ones rather than



Platform Independence in Java



ends on the

number of
acing such

Portability....

- ▶ At detail design level, the design of algorithms and data structures should not heavily depend on the platform specific feature so that portability can be obtained.

Portability....

- ▶ As for interface design, if an interface is heavily dependent on the environment specific features, the moving from one environment to another will be difficult.
- ▶ With the standardisation of graphic user interface supporting packages and their availability in various environments, interface designs are more or less environment independent.
- ▶ However, even if an interface is represented in an environment independent standard format such as in HTML, different web browsers may give different results and cause problems in using the same software in different environments.

Please explore

► <https://dotnet.microsoft.com/en-us/app>

.NET Multi-platform App UI

Build native, cross-platform desktop and mobile apps all in one framework.

Get started

Read docs

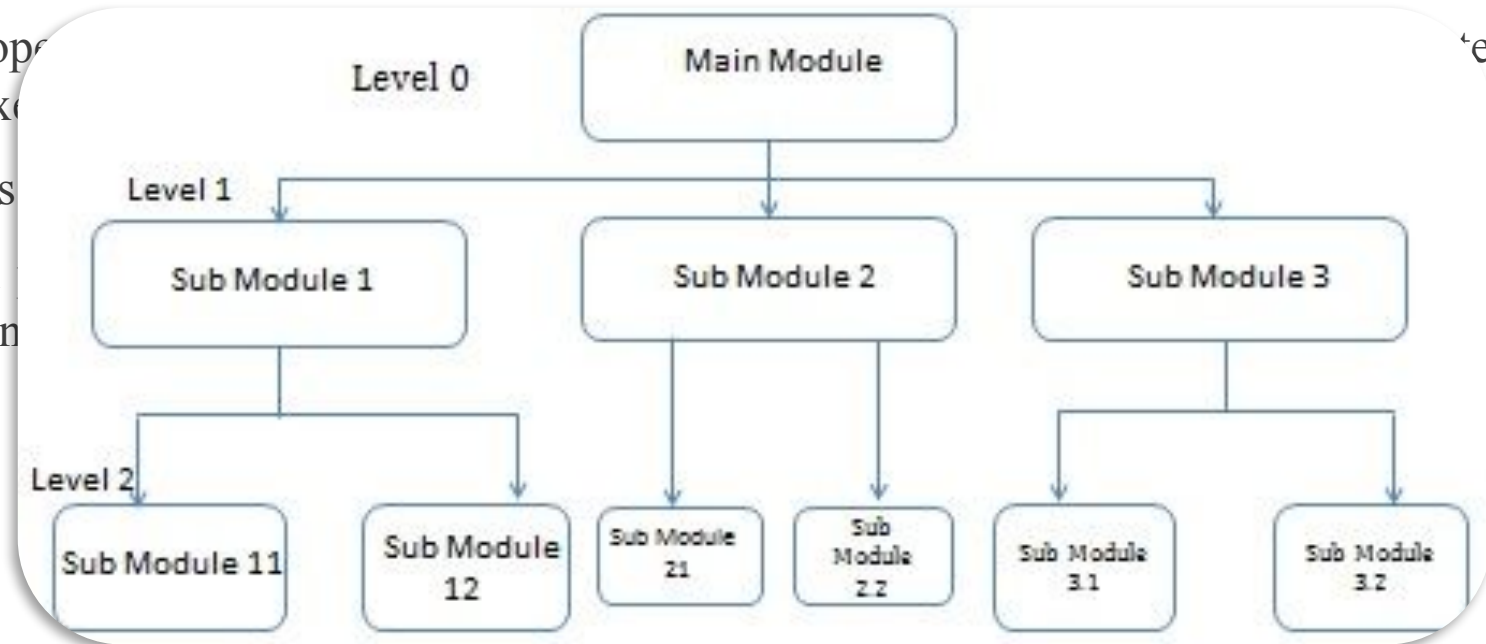


Maintainability

- ▶ Maintainability refers to the easiness of maintaining a software system. There are two types of software maintenance operations.
- ▶ One is the modification of the software system **for correction of bugs** that are found during the operation. Such modifications are called corrective maintenance.
- ▶ The second is the modification of the system due to **environment changes**, such as the upgrade of the system software, e.g. the operating system and the database management system. Such modifications are usually called adaptive maintenance.

Maintainability.....

- ▶ Both types of maintenance operations work so that bugs can be fixed
- ▶ Well-structured design helps
- ▶ Hence, architectural design is dependent on detail design and



tem

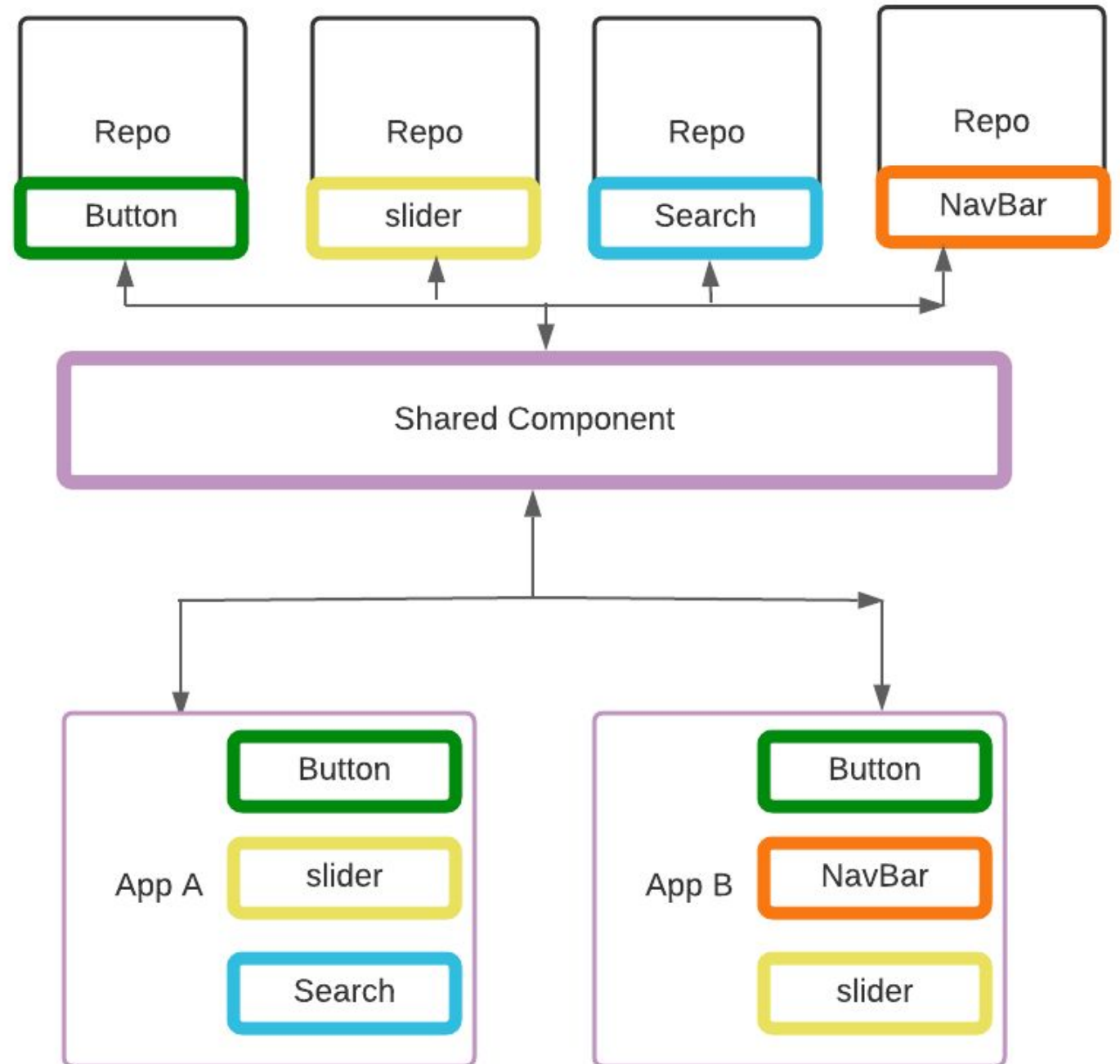
ly

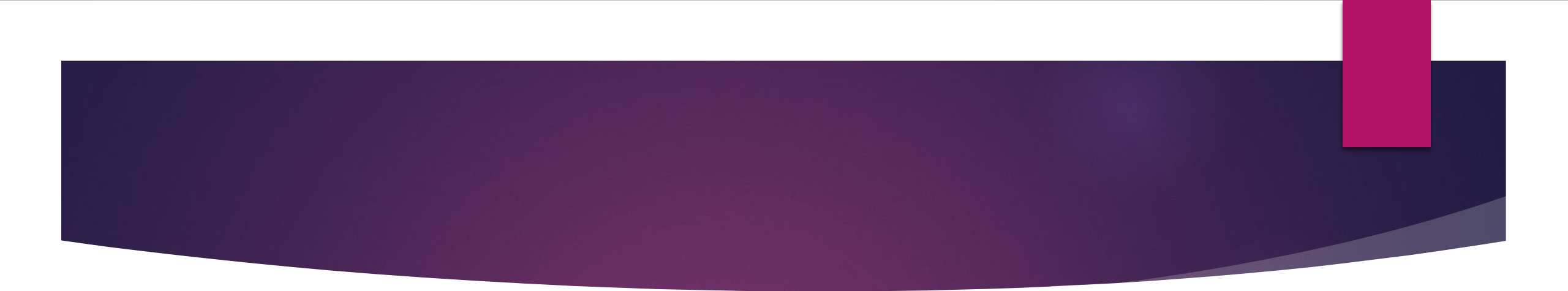
Reusability

- ▶ Reusability is the property of a software system that its components can be easily reused in the development of other software systems.
- ▶ Reusability depends on the generality of the components in a given application domain and the extent to which the components are parameterised and configurable



- ▶ Architecture design is obviously of si
a software system is decomposed into
- ▶ If such architecture can be used in a s
be developed for reuse again and again



- 
- ▶ Detailed design in terms of algorithm and data structure design also play a role in reusability in the way that they determine how easily the components can be parameterised and the way to configure the components.
 - ▶ However, they are less influential to reusability.
 - ▶ Interface design contributes to the reusability in a minor way in that it determines how reusable the interface is.

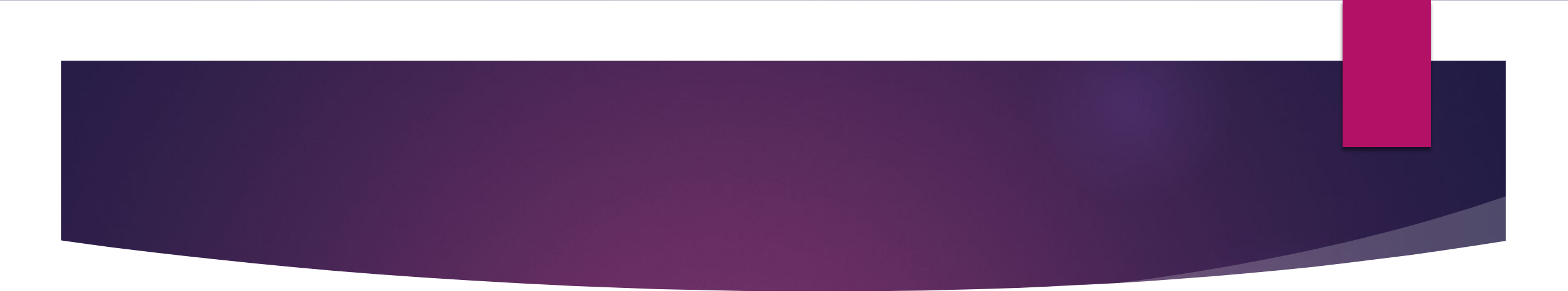
Interoperability

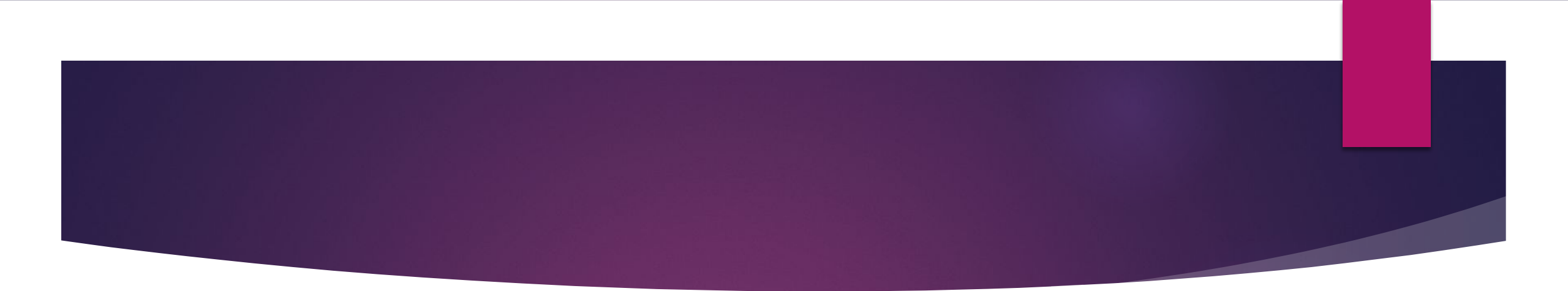
- ▶ Interoperability is the property of how easy a software system can be used with other software systems.
- ▶ For example, the interoperation of a word processor with an image processing system requires that the result of one software system, such as image processing of a photo, can be used in another so that a document can contain both text and image contents

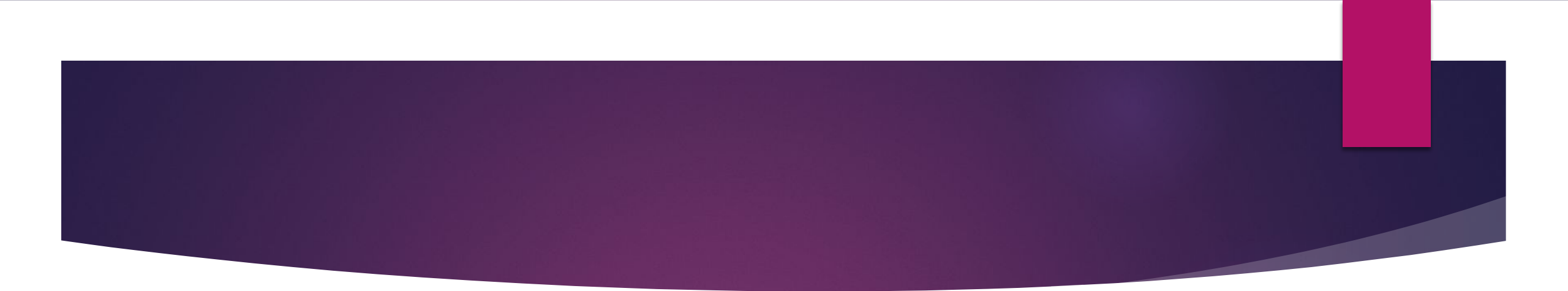
Interoperability....

- ▶ Interoperability mostly depends on the interface between a software system and its environment.
- ▶ It requires that the implementation of the software follow certain standard interface and coding conventions.
- ▶ Architecture design and algorithm and data structure design have little to do with interoperability.

Summary

- 
- ▶ **Effects of design on software quality demonstrates that design is critical to many of the quality attributes of software systems, and these qualities should be designed in and evaluated at the design phase.**
 - ▶ However, some quality attributes are not sensitive to certain levels of design. It would not be fruitful attempting to achieve qualities on these aspects via design or to analyse a design for its quality on such aspects. It should be noticed that the above discussion is not conclusive because it may be not true for all software systems.

- 
- ▶ In terms of different levels of design, algorithms and data structures are crucial to the correctness of the implementation of the functionalities of a software system.
 - ▶ The structural issues include the division of functions into components, the interfaces between the components and the distribution of software components onto physical computational resources.
 - ▶ These issues are crucial to almost all software quality attributes. This is especially true of efficiency, reusability, portability, interoperability, maintainability, and reliability, etc.

- 
- ▶ Interface between a software system and its environment, including the hardware and software platform that it executes on, is crucial to the portability.
 - ▶ Human computer interaction design is crucial to the usability of the system and may have a significant impact on human errors in the use of the system

2. اَللّٰهُمَّ صَلِّ عَلٰى مُحَمَّدٍ وَعَلٰى آلِ مُحَمَّدٍ كَمَا
صَلَّيْتَ عَلٰى اِبْرَاهِيْمَ وَعَلٰى آلِ اِبْرَاهِيْمَ اِنَّكَ حَمِيْدٌ
مَّجِيْدٌ اَللّٰهُمَّ بَارِكْ عَلٰى مُحَمَّدٍ وَعَلٰى آلِ مُحَمَّدٍ كَمَا
بَارَكْتَ عَلٰى اِبْرَاهِيْمَ وَعَلٰى آلِ اِبْرَاهِيْمَ اِنَّكَ
حَمِيْدٌ مَّجِيْدٌ.

اے اللہ! اپنی رحمت نازل فرما محمد پر اور آل محمد پر جیسا کہ تو نے اپنی رحمت نازل فرمائی ابراہیم پر اور آل
ابراہیم پر۔ بیشک تو تعریف والا اور بزرگی والا ہے۔ اے اللہ! برکت نازل فرما محمد پر اور آل محمد پر جیسا
کہ تو نے برکت نازل فرمائی ابراہیم پر اور آل ابراہیم پر۔ بیشک تو تعریف والا اور بزرگی والا ہے۔