Q.1 What is an Architecture ? where do architecture come from ? Importance of architecture ?

→

- Architecture has emerged as a crucial part of design process.
- Software architecture encompasses structure of large s/w systems.
- Architectural view of the system is abstract, distiding away details of implementation, algorithm, data representation and concentrating an behaviour and interaction of "black box" testing elements.
- s/w architecture is developed as first step towards designing a system that has collection of desired properties.

→ Where do architecture come from ?

- An architecture is result of a set of a business and technical decisions. There are many influences at work in its design and the realization of these influences at work in its design & realization of these influences change depending an environment which architecture is required to perform.
- An architecture designing a system for which real time deadlines are believed to be tight will make one set of design choices. Some architect designing a similar system in which deadlines can be easily satisfied, will make different choices.
- And some architect designing a non-real time system is likely to make quite different choices as well.

→ Importance of Architecture:

There are fundamentally three reasons for importance of s/w architecture.

1) Commⁿ among Stakeholders:

- s/w architecture represents a common abstraction of a system that most if not all of systems stakeholders can use basis for mutual understanding, negotiation, and communications

2) Early Design Decision:

- s/w architecture manifest earlier design decision about a system, and these early binding carry weight for out of propagation to their individual gravity with respect to system item. development, deployment and maintenance life cycle.

3) Transferable abstraction of a system:

- s/w architecture constituted of a reliability small, intellectually model for how a system is structured and how its elements work together and this model is transferable across system.

Q.2 Explain the factors which influences the architecture?

→

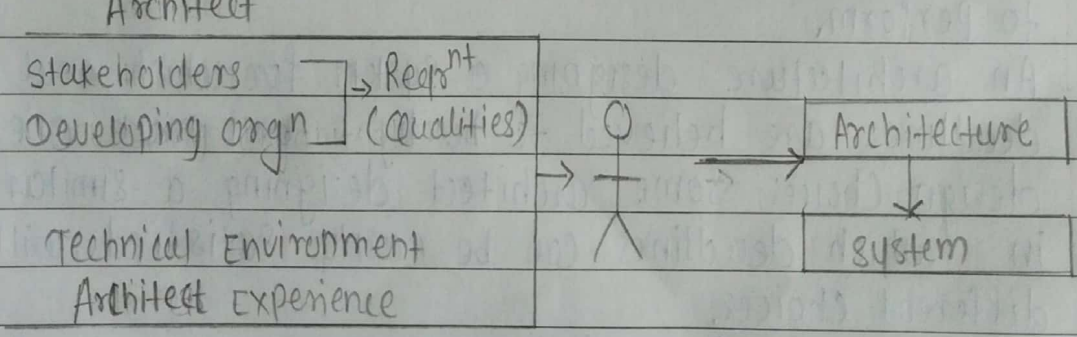

fig. Influences on Architecture

Page No.:
Date:
YOUVA

- Architectures are influenced by:
1) System stakeholders
2) Developing organization.
3) Background and experience of architects
4) Technical Environment.

→ 1) Stakeholders:
- Many people and organization are interested in Constructions of a s/w system. we call these stakeholders. The customers, end users, the developers, project manager, maintainers, and even those who market system are few examples.
- Stakeholders have different Concern and wish the system to guarantee or optimize, performing well on particular piece of h/w, easy to customize, low cost of development.

2) Developing organization:
- In addition to organization goals expressed through required architecture is influenced by structure of the nature of the development organization.

3) Background & experience of architects:
- If architects for system would have had good results using a particular architectural approach, such as the distributed objects or implicit invocation, Chances are that they will try same approach on new development effort.

4) Technical Environment:
- Special Case of architects background & experience is reflected by technical environment. Environment that in current when architecture is designed will influence that architecture.
- It might include an industry practises or s/w engg. techniques prevalent in architects, professional Community. It is brave architect who in ~~no~~ Todays environment does not least Consider a web based, object-oriented, middleware supported

design for information system.

Q. Explain software process and Architecture Business cycle.

→

- software process is term given to organization and management of SW development activities.
- These activities include -
1) Creating business case for system.
2) Understanding requirements.
3) Creating ~~or~~ selecting the architecture.
4) Documenting the architecture.
5) Analyzing or evaluating the architecture.
6) Implementing system based on architecture.
7) Ensuring that the implementation confirms to architecture.
- As indicated in structure of ABC, architecture activities compress to ensure feedback relationships with each other.
- They are as follows:

1) Creating Business case system:

- Creating a business case is broader than simply accessing the market need for a system. It is an important step in Creating and constrainting any future requirement. How much should the product cost? what in its targeted market what is its targeted time to market & will it need to interface with other systems?

2) Understanding Requirement:

- There are a variety of techniques for eliciting requirement from stakeholders. For e.g. object oriented analysis uses scenarios, or "Uselases" to embody requirements.
- One fundamental decision with respect to system being built is the event to which it is a ~~variation~~ variation or

Page No.:
Date:
YOUVA

other systems that have been constructed.

3) Communicating the architecture:
- For architecture to be effective as back bone of project design, its must be communicated unambigiously to all of stakeholders. Developers must understand the work assignment it requires of them, testers must understand task structure it imposes on them.
- Towards this end, architecture documentation should be informative, unambigious, and readable by many people.

4) Analyzing or evaluating the Architecture:
- Evaluating an architecture for the qualities that is supports is essential to ensure that the system constructed from that architecture satisfies it's stakeholders needs. Becoming more widespread are analysis techniques to evaluate the quality attributes that an architecture imparts to a system. Scenario based techniques provide one of the most general and effective approaches for evaluating an architecture.
- The most mature methodological approach is found in ATAM & CBAM.

5) Implementing system based on Architecture:
- This activity is concerned with keeping the developers faithful to the structures and interaction protocols constrained by the architecture. Having an explicit and well-communicated architecture is the first step toward ensuring architectural conformance. Having an environment or infrastructure that actively assists developers in creating and maintaining the architecture (as opposed to just the code) is better.

Page No.:
Date:
YOUVA

7) Ensuring Conformance to an Architecture:

- When an architecture is created and used, it goes into a maintenance phase. Constant vigilance is required to ensure that the actual architecture and it's representation remain faithful to each other during this phase. Although work in this area is comparatively immature, there has been intense activity in recent years.

Q. How architecture affect the factors that influence them?

→

- Some of the feedback comes from the architecture itself, and some comes from the system built from it.

Here is how the cycle works.

1. The architecture affects the structure of the developing organization. An architecture prescribes a structure for a system; as well as see, it particularly prescribes the units of software that must be implemented (or otherwise obtained) and integrated to form the system. These units are the basis for the development project's structure. Teams are formed for individual software units; and the development, test, and integration activities all revolve around the units. Likewise, schedules and budgets allocate resources in chunks corresponding to the units. If a company becomes adept at building families of similar system, it will tend to invest in each team by nurturing each area of expertise. Teams become embedded in the organization's structure. This is feedback from the architecture to the developing organization.

Page No.:
Date:
YOUVA

2. The architecture can affect the goals of the developing organization. A successful system built from it can enable a company to establish a foothold in a particular market area. The architecture can provide opportunities for the efficient production and deployment of similar systems, and the organization may adjust it's goals to take advantage of its newfound expertise to plumb the market.

3. The architecture can affect customer requirements for the next system by giving the customer the opportunity to receive a system (based on the same architecture) in a more reliable, timely, and economical manner than if the subsequent system were to be built from scratch. The customer may be willing to relax some requirements to gain these economies.

4. The process of system building will affect the architect's experience with subsequent systems by adding to the corporate experience base. A system that was successfully built around a tool bus or .NET or encapsulated finite-state machines will engender similar systems built the same way in the future. On the other hand, architectures that fail are less likely to be chosen for future projects.

5. A few systems will influence and actually change the software engineering culture, that is, the technical environment in which system builder operate and learn. The first relational databases, compiler generators, and table-driven operating system had this effect in the 1960s and early 1970s; the first spreadsheet & windowing systems, in the 1980s.

Q. Explain ABC cycle in brief with diagram?

→

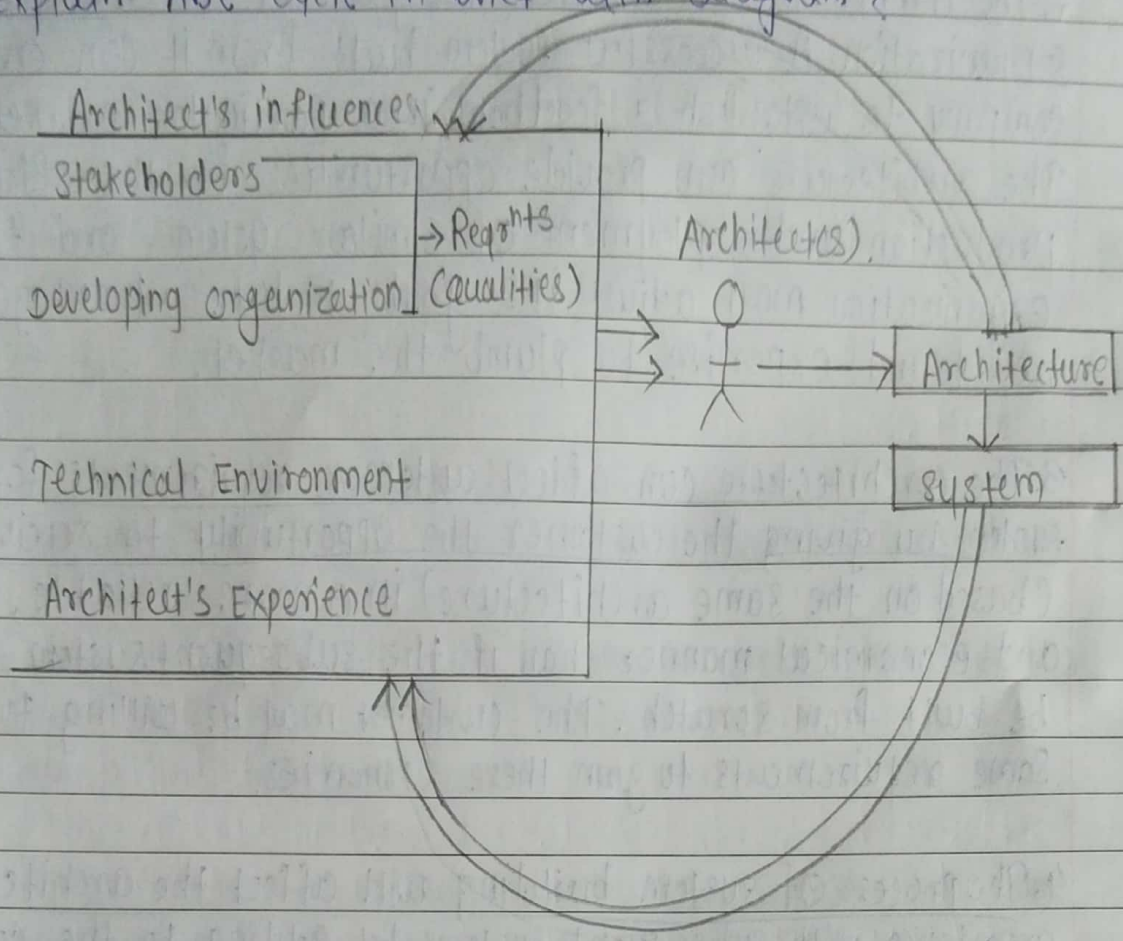

Fig. The Architecture Business cycle.

- A business manages this cycle to handle growth, to expand its enterprise area, and to take advantage of previous investments in architecture & system building.
- Above figure shows the feedback loops. Some of the feedback comes from the architecture itself, & some comes from the system built from it.
- Here is how the cycle works:

(1.) The architecture affects the structure of the developing organization. An architecture prescribes a structure for a systems,

Page No.:
Date:
YOUVA

2. The architecture can affect goals of the developing organization.
3. The architecture can affects customer requirements for the next system by giving the customer the opportunity to receive a system in a more reliable, timely & economical manner than if the subsequent system were to be built from scratch.
4. The process of system building will affect the architect's experience with subsequent systems by adding to the corporate experience base.
5. A few system will influence and actually change the SW engineering culture. That is, the technical environment in which system builders operate & learn.

- These and other feedback mechanism form what we call the ABC, illustrated in above figure, which depicts the influences of the culture and business of the development organization on the SW architecture.
- That arch. is, in turn, a primary determinant of the properties of the developed system or systems. But the ABC is also based on a recognition that shrewd organizations can take advantage of the organizational & experiential effects of developing an arch. and can use those effects to position their business strategically for future projects.

Q. What makes a "Good" Architecture?

→

- If it is true that, given the same technical requirements for a system, two different architects in different organizations will produce different architectures, how can we determine if either one of them is the right one?

Page No.:
Date:
YOUVA

- There is no such thing as an inherently good or bad architecture. Arch. are either more or less fit for some stated purpose. A distributed three-tier client-server arch. may be just the ticket for a large enterprise's financial management system but completely wrong for an avionics application.

- We divide our observations into two clusters: Process recommendations and product (or structural) recommendations. Our process recommendations are as follows:

- The architecture should be the product of a single architect or a small group of architects with an identified leader.
- The architect (or arch. team) should have the functional requirements for the system and an articulated, prioritized list of quality attributes (such as security or modifiability) that the arch. is expected to satisfy.
- The architecture should be well documented, with at least one static view and one dynamic view
- The arch. should be circulated to the system's stakeholders, who should be actively involved in it's review.

→ Our structural rules of thumb are as follows:

- The architecture should feature well-defined modules whose functional responsibilities are allocated on the principles of info^n hiding & separation of concerns.

- Each module should have a well-defined interface that encapsulates or "hides" changeable aspects (such as implementation strategies and data structure choices) from other s/w that uses it's facilities
- The arch. should never depend on a particular version of a commercial product or tool.
- Modules that produce data should be separate from modules that consume data.

Q. what software is and what isn't?

→

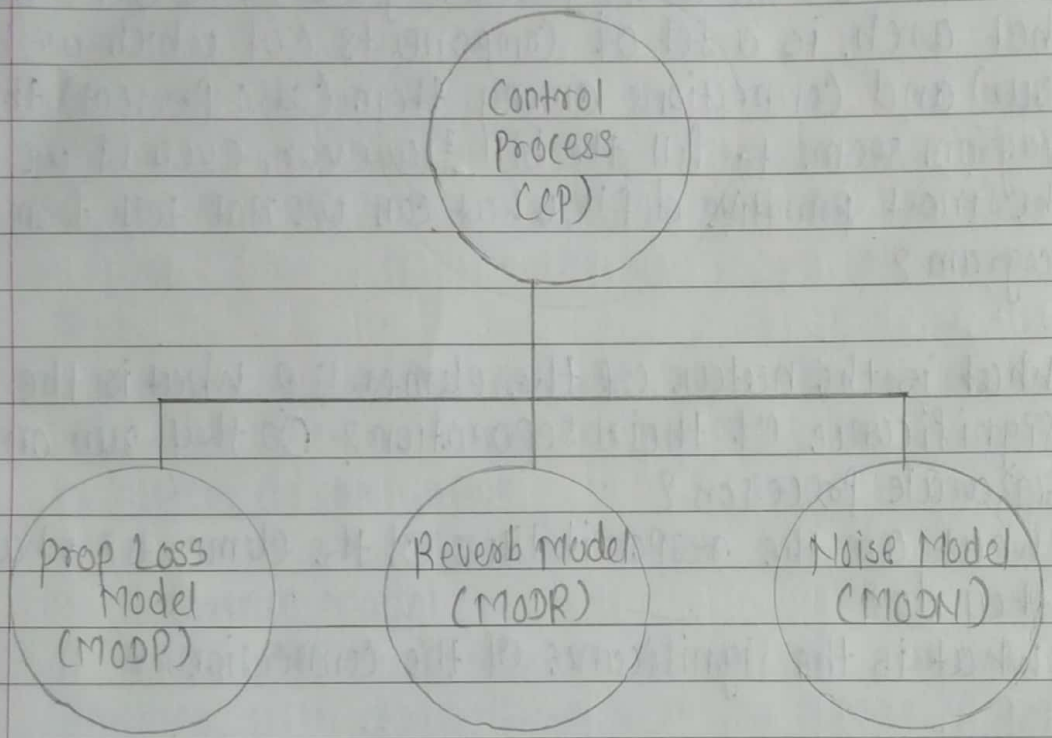

fig. Presentation of a s/w architecture.

- above figer, taken from a system description for an underwater acoustic simulation, purports to describe that system's "top-level architecture" and is precisely the kind of diagram most often displayed to help explain an architecture. Exactly what can we tell from it?

- The system consists of four elements.
- Three of the elements, prop loss Model (MODP), Reverb Model (MODR), and Noise Model (MODN), might have more in common with each other than with the fourth, Control Process (CP), because they are positioned next to each other.
- All of the elements apparently have some sort of relationship with each other, since the diagram is fully connected.

- Is this an architecture? Assuming (as many defns do) that arch. is a set of Components (of which we have four) and Connections among them (also present), this diagram seems to fill the bill. However, even if we accept the most primitive defn, what can we not tell from the diagram?

  - what is the nature of the elements? what is the significance of their separation? Do they run on separate processors?
  - what are the responsibilities of the elements? what is it they do?
  - what is the significance of the connections?

- This dia. does not show a s/w arch., at least not in any useful way. The most charitable thing we can say about such dia. is that they represent a start. We now define what does constitute a s/w arch.

Q. Define Architectural patterns, Reference Models, Reference Architecture?

Page No.:
Date:
YOUVA

→

→ 1) Architectural pattern-

- An architectural pattern is a description of element and relation types together with a set of constraints on how they may be used. A pattern can be thought of as a set of constraints on an architecture - on the element types and their patterns of interactions. and these constraints define a set or family of architectures that satisfy them.
- e.g. Client - server is a common architectural pattern
- one of the most useful aspects of patterns is that they exhibit known quality attributes.

→ 2) Reference Architecture:

- Reference arch. is a reference model mapped onto SW elements (that cooperatively implement the functionality defined in the reference model) and the data flows betn them.
- whereas a reference model divides the functionality, a reference arch. is the mapping of that functionality onto a system decomposition.

→ 3) Reference Model:

- A Reference model is a division of functionality together with data flow betn the pieces. A reference model is a standard decomposition of a known problem into parts that cooperatively solve the problem.
- Arising from experience, reference model are a characteristic of mature domain.

→ Reference models, architectural patterns & reference architectures are not architectures; they are useful

Concepts that capture elements of an architecture

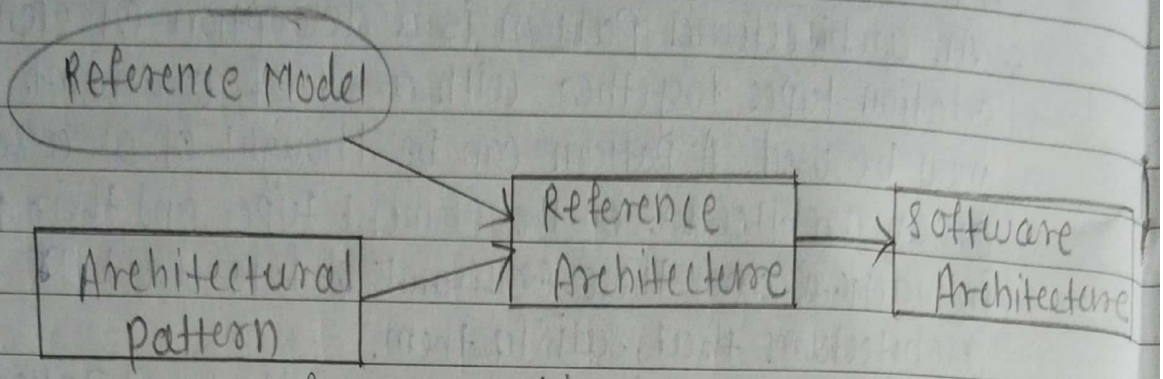

fig. SIW architecture.

Q. What is architectural structure and views?

→

→ A view -

- A view is a representation of a coherent set of architectural elements, as written by and read by system stakeholders.
- It consists of representation of a set of elements and the relations among them.

→ Structure.

- A structure is the set of elements itself. as they exist in software or hardware.
- e.g. A module structure is the set of the system's modules and their organization.
- Architectural structures can by and large be divided into three groups, depending on the broad nature of the elements they show.

- Module structure
- Component and connector structures
- Allocation structures.
- Software structures.

Q. Explain ABC cycle for A-7E AVIONIC system?

→

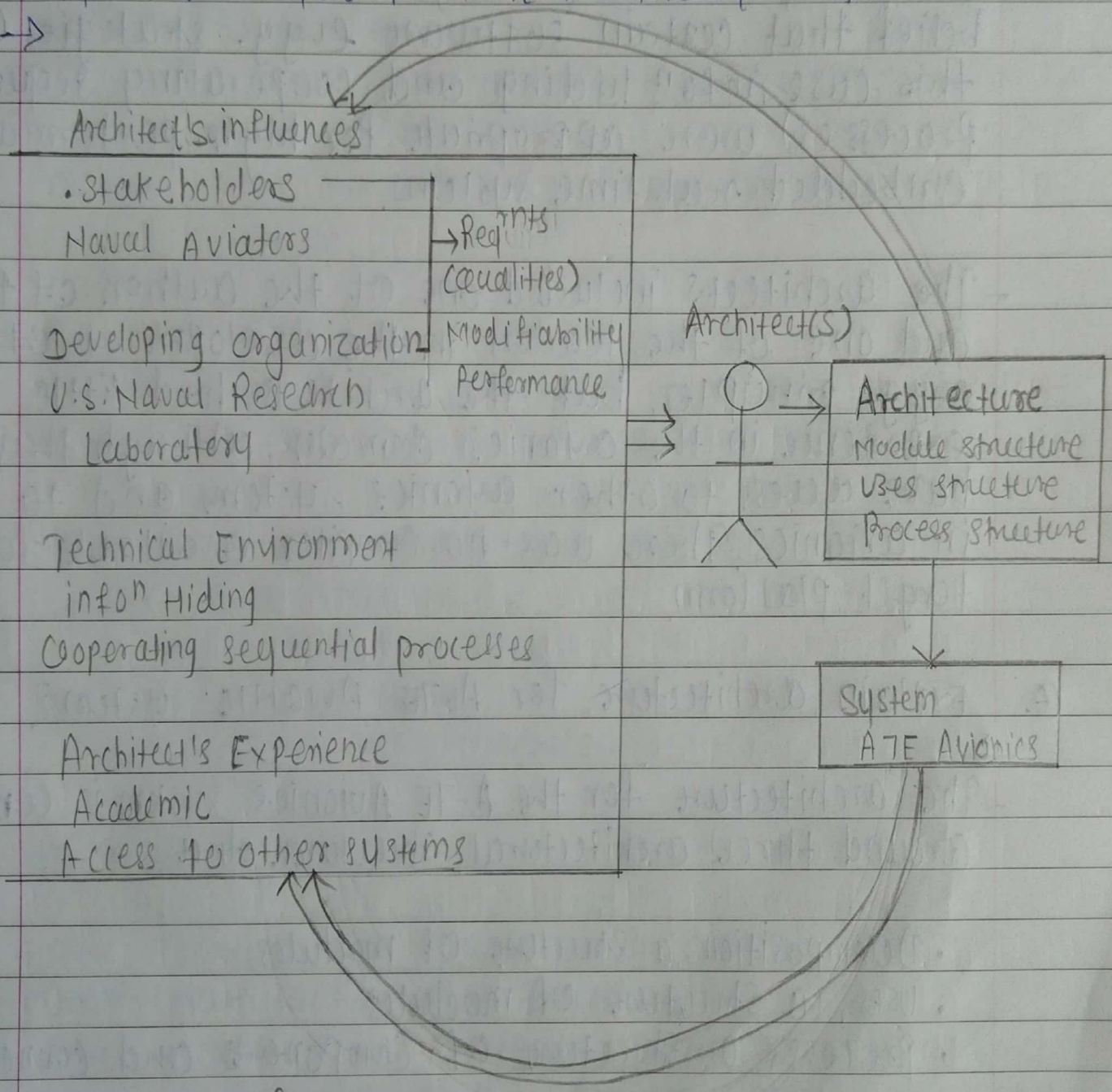

fig. The ABC cycle for A-7E AVIONIC systems.

- Above figure shows the ABC as it pertains to the A-7E

Page No.:
Date:
YOUVA

Avionics system described in this ch cycle. The system was constructed begining in 1971 for the naval aviators who flew the A-7E aircraft and was paid for by the U.S. Navy.

- The developers were creating the SW to test their belief that certain software engg. strategies (in this case, info hiding and cooperating sequential processes) were appropriate for high-performance embedded real-time systems.

- The architects included one of the authors of the book and one of the leaders in the development of SW engg. principles, but the architects had little experience in the avionics domain, although they did have access to other avionics systems and to experts in avionics. There was no compiler available for the target platform.

Q. Explain architecture for A-7E AVIONIC system?

→

- The architecture for the A-7E Avionics system is centered around three architectural structures, viset

  - Decomposition, a structure of modules
  - Uses, a structure of modules
  - Process, a structure of components and connectors

→ 1) Decomposition Structure:

- Unless a program is small enough to be produced

Page No.:
Date:
YOUVA

by a single programmer, we must think how the work will be divided into units that can be implemented separately and how those modules will interact.
- The unit of decomposition structure is, of course, the module.

→ Information Hiding.
- The A-7E module decomposition is based on information hiding. info^{n} hiding works by encapsulating system details that are likely to change independently in diff^{nt} modules.

→ A-7E module Decomposition structure:
- To describe the A-7E module decomposition structure, and to give an example of how a module structure is documented, we provide the following excepts from the A-7E S/W module guide
  - H/W-Hiding Module.
  - Behavior-Hiding Module.
  - S/W Decision Module.
  - Appl^{n} Data type module.

→ 2) Uses structure:

- The second major structure of interest in the A-7E architecture is the uses structure.
- The decomposition structure carries no info^{n} about runtime execution of the software, you might make an educated guess as to how two procedures in diff^{nt} modules interact at runtime, but this info^{n} is not in fact in the module decomposition. Rather, the

Page No.:
Date:
YOUVA

Uses structure supplies the authoritative picture of how the SW interacts.

→ The A-7E Uses structure:

- Recall that the uses structure is first documented in a specification showing the allowed-to-use relation: actual uses are extracted after implementation.

→ 3) Process structure:

- The third structure of architectural importance to the A-7E is the process structure. Even though the underlying aircraft computer is a uniprocessor, the Extended Computer module presents a virtual programming interface that features multiprocessing capabilities.
- This was to plan for if and when the A-7E computer was replaced with an actual multiprocessor.
- A process is a set of programming steps that are repeated in response to a triggering event or to a timing constraint.

- The process structure emerged after the other structure had been designed. Function Driver procedures were implemented as processes.
- Other processes computed time-consuming calculations in the background so that a value would always be available.

Q. Describe requirement and quality factor for A-7E AVIONIC System ?

→

- An earlier version, the A-7C, was among the very first production aircraft in the world to be equiped with an onboard computer to help the pilot with navigation & "weapon delivery".

- The A-7E's onboard computer is a small, special-purpose IBM machine for which no compiler exists; programming is in assembly language only. The computer has special registers connected to analog-to-digital and digital-to-analog converters that let it receive and send data to almost two dozen devices in the aircraft's avionics suite.

- In broad terms, the A-7E software is responsible for reading sensors and updating cockpit displays that help the pilot drop weapons on a target. The A-7E SIW does not actually fly the aircraft, as more modern avionic systems do.

→ The following are the primary sensors the SIW reads and manages:

- An air probe that measures barometric pressure and air speed.
- A forward-looking radar that can be aimed in azimuth and elevation and returns the straight-line range to the point on the ground at which it is pointed.
- A radar alimeter that measures the distance to the ground.