**Experiment No.: 8 Identifying System requirements for an Architecture for any specific domain.**

**Learning Objective:** Student should be able to understand System requirements for an Architecture for any specific domain**.**

**Theory:**

The purpose of a requirements architecture is to structure and organize requirements in such a way that the requirements are stable, usable, adapt to changes, and are elegant When a requirements architecture is faulty, it can cause problems. When the requirements architecture is poor, the following problems result:

1. No one knows why a requirement was changed
2. Requirements cannot be reused
3. Traceability is superficial or unused by other teams
4. Requirements reviews involve irrelevant information
5. Big picture of the system being built and reasons for building it are not well-understood
6. It is important to keep in mind that the purpose of a good requirements architecture is to build working software that meets business objectives.

Software requirements must be testable, unambiguous, and concise, a requirements architecture must also possess certain attributes. The above blueprint provides some general guidelines for how to structure requirements, but keeping in mind the following attributes:

1. **Maintainable**: Whatever choices you make in organizing requirements, ensure that you create a structure that can adapt to changes in requirements.
2. **Traceable**: Do you know which requirements any given process flow step is traced to?
3. **Usable**: Consider the stakeholders in the org chart—are the requirements architected in such a way that you could either produce output for each of them or such that they could navigate to the requirements in the tool and find the requirements objects that are relevant to them? The hierarchies and traces you create should be consistent: Don’t create one hierarchy where the FRs are children of the models and another hierarchy for the same project where FRs are not children of the models but are traced to them.
4. **Scalable**: Imagine your requirements architecture with 10 times the number of requirements it has. Now imagine it with 100 times the number of requirements. Architectures should be able to support the addition of new requirements with minimal overhead.
5. **Elegant**: Are there just enough hierarchies to facilitate use? Are you repeating hierarchies just to make traceability easier? Does your architecture contain duplicate models or requirements
6. **Generalizable**: The architecture approach should be repeatable. You ought to be able to go into any project and no matter the domain use the same approach to requirements architecture.

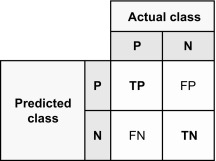
All architectures are tradeoffs – like in software architecture, you may need to sometimes sacrifice aesthetics for robust traceability or reuse.  Domain-specific Software Development Various mechanisms of domain-specific software development are under investigation within the projects. Following are the system requirement of Domain-specific Software Development:

* Performance
  + How quickly must the system respond to interactive operations of different kinds?
  + Are there different classes of interactive operations that users have different tolerances / expectations for?
  + Is there a batch window? What runs in it?
  + Do the batches have their own performance constraints, e.g., to clear the batch window before it closes?
  + Does the batch load influence any interactive users running at the same time?
  + Is there data with a high read/write access ratio that can be cached in memory at different tiers in the architecture?
  + What are the expected performance bottlenecks?
    - CPU?
    - Memory on client, server or intermediate nodes?
    - Hard drive space on each node?
    - Communications links?
    - DB-Access, Searching, Complex joins
    - Interaction with partner systems?
    - Interactions with public systems?
* Scalability
  + Peak load of how many users doing what kinds of operations?
  + Ability to grow to how many records in which critical database tables without slowing down related operations by more than X
  + Avoiding saturating a communication link that cannot be upgraded to a higher speed?
  + What dimensions can be scaled, e.g., more CPUs, more memory, more servers, geographical distribution?
* Availability
  + What is the required uptime percentage?
  + Does this vary by time of day or location?
  + What is the current schedule of controlled outages? Is this acceptable, or is there a goal to improve it?
* Reliability
  + Are there components with re-liabilities that are known to be less than the required reliability of the system?
  + What strategies are currently in place to build more reliable capabilities
  + What is the expected mean time to failure by failure severity by operation?
  + How will reliability be assessed prior to deployment?
* Security
  + What operations need to be secured?
  + How will users be administered?
  + How will users be given permissions to access secured operations?
  + What are the different levels of security and how do these map-Security by operation, Security by type of object, Security by instance of object
* Maintainability
  + Are there concerns about the ability to hire appropriate technology skills, attract them to the area at reasonable prices?
  + What kinds of changes are anticipated in the first rounds of maintenance? What are their relative priority?
  + What sort of regression testing is required to ensure that maintenance changes do not degrade existing functionality?
  + What sort of maintenance documentation is expected to be produced? When?
* Flexibility
  + Is there system behavior that needs to be changed regularly without program changes?
    - Can this be encoded in the database?
    - Are there run-time rules that can be handled using a rules interpretation engine?
    - Are there functions that should be user scripted? If so, how will these be QA-ed?
* Configurability
  + What parameters need to be set on a machine-by-machine basis?
* Personalizability
  + What aspects of the system can be customized on a per-user basis?
  + How does the user change these settings?
  + What is the strategy for defaults?
* Usability
  + Are there operations that need to be done as quickly as possible, so that user gestures should be minimized?>
  + Are there difficult or occasional-user operations that require non-standard presentations to help the user perform correctly?
  + What is the balance between data integrity and the ability to stop in a "work in progress" state?
  + What metaphors from existing or parallel systems should be used?
  + What sort of training deliverables are expected?
  + What sort of on-board help system is expected?
* Portability
  + Data portability between this system and other systems?
  + Portability across different versions of a single vendor's DB?
  + Ability to port to a different vendor's DB? Which one(s)? When?
  + Browser portability? What browser versions? Historical and future?
  + Operating system portability?
* Conformance to standards
  + What legal standards apply?
  + What technical standards apply?
  + Other standards, e.g., 508.1 for disabled users?
  + What development standards apply?
    - Database naming standards
    - Existing internal architectural standards (e.g., everything goes in an Oracle database)
    - Language and coding standards
    - Testing and review standards
    - Presentation standards, e.g., use of standard colors, controls or other affordances?
    - Lifecycle models or methodologies
* Internationalizability
  + What languages?
  + In what order?
  + How translated?
  + Single or multi byte character sets?
* Efficiency -- space and time
* Responsiveness
  + What are the expected and upper limit response times per operation in the system?
  + What is the trade-off between lower averages and wider variations in response time?
* Interoperability
  + What systems will this system interoperate with immediately?
  + What other systems are anticipated?
  + What classes of internal and external systems might later be needed to interoperate with?
  + What functionality from this system needs to be exposed as a service in a service oriented architecture?
  + What functionality from this system needs to be exposed as a Web service or via a portal?
* Upgradeability
  + Do the servers need to be upgraded while running?
  + How many client stations need to be upgraded, and what are the costs and mechanisms for upgrading them?
  + How often do different kind of fixes need to be distributed? Are there "hot fixes" that have to go out right away, but others that can wait? How often do each kind occur?
* Auditability / traceability
  + What record of who did what when must be maintained?
  + For how long?
  + Who accesses the audit trails? How?
  + Is archive to tape or other off-site storage media required?
  + Is "effective dating" required?
* Transactionality
  + What are the important database and application transaction boundaries?
  + Is standard "optimistic" locking appropriate, or is something more complex required in some or all cases>. Is disconnected operation required by any node?
* Administrability
  + What live usage information needs to be displayed?
  + To who? How? When? What "live" interventions are required?
  + What ability to handle remote configurations are required?
  + Are there existing application management consoles that will be used to manage this application?

**Result and Discussion:**

**Confusion matrix:**

A confusion matrix represents the prediction summary in matrix form. It shows how many prediction are correct and incorrect per class. It helps in understanding the classes that are being confused by model as other class.



**Precision:**

Precision is a statistical measure used in various fields, including machine learning, statistics, and quality control, to assess the accuracy of a model or system's positive predictions or classifications. It quantifies the proportion of true positive predictions or correct positive classifications among all positive predictions made by the model.

The precision formula is typically expressed as:

**Precision = True Positives / (True Positives + False Positives)**

**Precision = TP / (TP + FP)**

**Recall:**

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

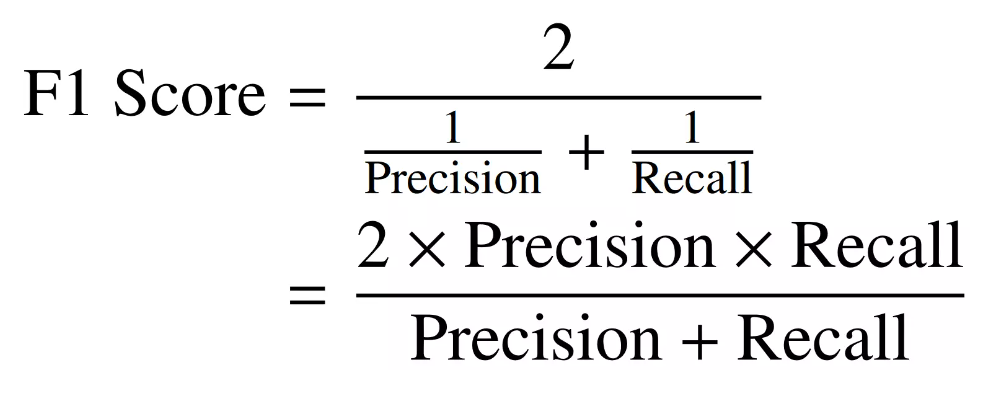
**Recall = True Positive/True Positive + False Negative**

**Recall = TP/TP+FN**

**F1 Score:**

The F-score, also called the F1-score, is a measure of a model’s accuracy on a dataset. It is used to evaluate binary classification systems, which classify examples into ‘positive’ or ‘negative’.

The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model’s precision and recall.



|  |  |  |
| --- | --- | --- |
| **System** | **Algorithm** | **Accuracy** |
| Cancer Prediction | Logistic Regression | 96.49 |
| Heart Disease Prediction | Logistic Regression | 80.48 |
| Diabetes Prediction | Random Forest Classifier | 90.78 |

**Learning Outcomes:** Students should have be able to understand

LO1: Define software System.

LO2: Identify different system requirements of software Architecture.

LO3: Explain system requirement of Domain-specific Software Development.

**Course Outcomes:** Upon completion of the course students will be able to understand System requirements for an Architecture for any specific domain.

**Conclusion:** Identifying system requirements for a domain-specific architecture is a crucial process that involves understanding domain intricacies, engaging stakeholders, and documenting prioritized functional and non-functional requirements. Effective communication and adaptation to changing needs are essential for a successful system design.

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** |  |
| **Marks Obtained** |  |  |  |