# G.Selected architectural styles and patterns:

Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system. In the model of the software development process, architectural design is the first stage in the software design process. It is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them. The output of the architectural design process is an architectural model that describes how the system is organized as a set of communicating components.

We choose agile processes, so we can empower the process of designing and developing Data4Help and AutomatedSOS , our target it was to apply the four important values of the agile manifesto that are :-

1- Focus should be more on individuals and interactions instead of processes and tools.

2-Working software is more important that comprehensive documentation.

3-Customer collaboration is more vital than contract negotiation.

4-The process should respond to change rather than follow a plan.

Not only but we took in consideration also the 12 principles of agile software development:-

1-Deliver customer satisfaction by delivering valuable software continuously.

2-Always accept change of requirements matter how early or late in the project.

3-Deliver software that works within a shorter timescale.

4-Both developers and business professionals must work closely together daily throughout the duration of the project.

5-Information is best transferred between parties in face-to-face conversations.

6-Motivate people to build a project by creating an environment of appreciation, trust, and empowerment.

7-Working software is the key measure of progress.

8-The agile process promotes sustainable development.

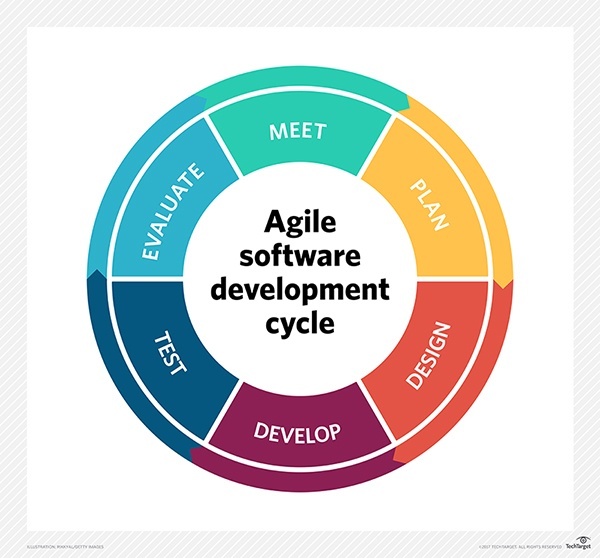
9-Continuous attention to excellence and quality in technical development and design boosts the agility.

10- Simplicity is a vital part of effective agile management.

11-Self-organized teams produce the best architecture, requirements, and design.

12-Teams should reflect through inspection and adaption to be more effective.

Also this figure describe the agile process: -



# Design Pattern

-**Repository pattern**

We choose Repository pattern because large amounts of data will be shared with TrackMe that will have a large repository that third party companies will access these data , we saw that repository pattern is the best choice for this project.

This table describes well the reason why we choose the repository pattern

|  |  |
| --- | --- |
| **Name** | **Repository** |
| **Description** | All data in a system is managed in a central repository that is accessible to all system components. Components do not interact directly, only through the repository. |
| **Why** | We use this pattern because the system have large volumes of information that need to be stored for a long time. |
| **Advantages** | Components can be independent—they do not need to know of the existence of other components. Changes made by one component can be propagated to all components. All data can be managed consistently (e.g., backups done at the same time) as it is all in one place. |
| **Disadvantages** | The repository is a single point of failure so problems in the repository affect the whole system. May be inefficiencies in organizing all communication through the repository. Distributing the repository across several computers may be difficult. |

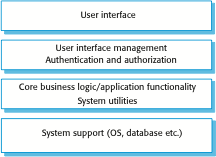
**-Layered pattern**

In addition to the repository pattern we also decided to use the layered architecture pattern to model the interfacing of sub-systems, the layered pattern also is useful in organizing the system into a set of layers (or abstract machines) each of which provide a set of services, and last but not least supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.

This is a table that describes well why we have chosen this pattern: -

|  |  |
| --- | --- |
| **Name** | **Layered architecture** |
| **Description** | Organizes the system into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system. |
| **Why** | We decided to use it when building new facilities on top of existing systems, to be more specific building AutomatedSOS on top of Data4Help, also it is useful when the development is spread across several teams with each team responsibility for a layer of functionality. |
| **Advantages** | Allows replacement of entire layers so long as the interface is maintained. Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system. |
| **Disadvantages** | In practice, providing a clean separation between layers is often difficult and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it. Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer. |

This is a picture that describes a generic layered pattern :-



Explaining this generic layered pattern and linking it with the software :-

**Layer 1** (User interface ) :- we have 3 user interfaces (TrackMe UI , Third party UI and User UI).

**Layer 2** (User interface management authentication and authorization ) :- In this layer each created account will have different authorities as described before in the RASD.

**Layer 3** (Core business logic/application functionality system utilities ) :- For example AutomatedSOS will calculate some data to send a request to the ambulance .

**Layer 4** (System support(OS, database etc.) ) :- We also mention before in the RASD that the mobile application will work on android platform and there will be a web application for both TrackMe and Third parties .