**Dizajnimi dhe modelimi i sistemve**

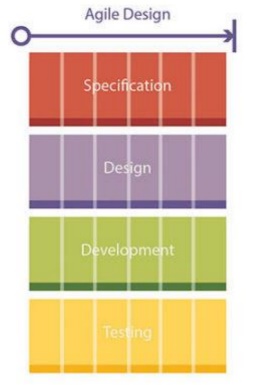
Në fazën e analizës së kërkesave ne arrijm të kuptojmë nevojat e biznesit, ndërsa Në fazën e dizajnimit dhe modelimit ne arrijm të kuptojmë se si ta duhet ta konfigurojmë dhe ndërtojmë sistemin ashtu që ti plotësoj ato nevoja. Në fazën e dizajnit e gjith puna logjike do të konvertohet në punë fizike.

Faza e dizajnit përfshin disa aktivitete të cilat asociohen me elementet përkatëse, më posht po i radhisë këto aktivitete : *Dizajnimi i ambientit* – Përfshin elemente si rrjetat, serverët, paisjet. *Dizajnimi i arkitekturës së softuerit* – Përfshin elementet si softuerin me komponentat dhe modulet e veta. *Dizajnimi i interface së sistemit* – Përfshin mjete për komunikimin me sistemet tjera (API). *Dizajnimi i ndërfaqes së përdoruesit* – Përfshin elemente si dritare, raporte. *Dizajnimi i data bazës* – Megazinimi i informacioneve. *Dizajnimi i kontrollit të qasjes dhe sigurisë* – Firewalls, kontrolli i qasjes.

Inxhinierët softuerik gjatë fazës së dizajnit e bëjn rishikimin e user stories së bashku me PO, për të identifikuar nevojat e në bazë të tyre të definojnë pastaj komponentat teknike të nevojshme që do duhet të jen pjesë e sistemit.

Ekipa gjatë kësaj faze merret me elemente si *integrimi i ariktekturës së sistemit, krijimi i ndërfaqeve të përdoruesit, krijimi i skemave të data bazës.*

**Elementet bazë të cilat duhet të dizajnohen në kët fazë janë** : *Arktiektura* – Është model konceptual që përcakton sjellien, pamjen dhe strukturën e sistemit, përdoren diagramet për ilustrimin dhe prezentimin e arkitekturës. *Modulet* – Janë pjesë të vogla(grup i funksioneve) të sistemit të cilat kryejn një detyr/operacion specifik përbrenda sistemit, modulet e formojnë sistemin. *Komponentet* - Janë grup i moduleve të cilat ndërlidhen për të kryer një funksion të caktuar. *Data Baza* – Menaxhimi i informacionit dhe rrjedhies së të dhënave. *Interface* – Janë kufi i përbashkët përmes të cilave formohen edhe vetë komponenetat, me anë të interface ndërlidhen modulet për të ofruar funksion të përbashkët.

Në agile fazat e SDLC i ndajmë në pjesë të vogla, të pa varura, të zbatueshme të cilat ecin në mënyr paralele dhe mundësojn dorëzim individual.

Në agile në fazën e dizajnimit bëhet dizajnimi i : *Arkitekturës* : Strukturë e përgjithshme e gjithë sistemi, duhet ndjekur paterna. *Modeli të domenit* : Bëjn pjesë dizajnimi i moduleve dhe klasave. *Use Casave Kryesorë* – Krijimi i use case diagrameve, diagrameve të aktivitetit etj.

*Secili vendim që mirren në fazën e dizajnit duhet të jet i rrjedhur nga kërkesat fuknsionale ose jo funksionale.*

Për të arritur sukses në procesin e dizajnit të softuerit duhet ndjekur/përmbushur këto veti/detyra : *1.Përdorimi* *i design patternave* si MVC, Layered, Hexagon, Microservices. *2.Dizajnimi i modelit të domenit të jet modular* dmth sistemit të ndahen në nën-sisteme ose nën-domaina. *3.Të dizajnohet databaza, komponentet dhe arkitektura. 4.Dizajnimi duhet të orientohet drejt strukturimit të të dhënave të përshtatshme për klasat të cilat implementohen dhe janë të varura nga paterni që e ndjekim. Krijimi i komponentave që kryejn funksione të pa varura. Krijim i Interface(API) të cilat e ulin kompleksitetin e komunikimit në mes interface. 5.Dizajnet duhet të rrjedhin nga kërkesat. 6.Dizjanet të prezenohen me anë të simboleve*.

**Paterni Layer** : Komponentat e sistemit i vendos si shtresa horizontale ku secila shtres e ka rolin e vet specifik brenda aplikacionit. Paterni nuk e kufizon numrin e shtresave mirpo 4 janë si numër standard i shtresave që duhet përfshirë, këtu hynë : *Shtresa e prezentimit, aplikacionit(shërbimet), qasja e të dhënave, integrimit.*

**Paterni MVC** : Aplikacionin e ndanë në 3 komponenta logjike M-*Model* V-*View* C-*Controller*. Model – *Modeli e përmban biznes logjikën dhe ndërvepron me data bazën*. View – *Gjeneronë UI që i shfaqet userit.* Controller – *Shërben si urë lidhse mes modelit dhe view, pranon requestat dhe kthen përgjigje tek useri.*

**Paterni Hexagonal** : Është alternativ e layer paternit, e përqëndron logjikën e biznesit (domenin) në qendër, quhet edhe si onion, clean architecture.

Aplikacioni **monolit** është i tëri i organizuar në një njesi të vetme, në tjetrën anë aplikacioni **microservice** është i ndarë në njesi të vogla të pa varura, të cilat komunikojn me njëra tjetrën duke përdorur metodat e quajtura **API**.

*Dizajnimi i GUI* i referohet dizajnimit të elementeve vizuale me të cilat përdoruesi ndërvepron me sistemin, në fakt për përdoruesin GUI paraqet të gjith sistemin, në dizajnimin e gui përfshihen elementet si *buttona, drop down lista, layout...*

*Dizajnimi i Data Bazës* – Përfshin krijimin e skemave, kufizimeve dhe diagrameve ERD. Ndahet në dy arkitektura : *Centralizuar* – I gjith aplikacioni përdor vetëm një data base dhe vetëm një server ku i ruan dhe i mer të dhënat nga. Është e lehtë për tu organizuar, kërkuara, backup dhe modifikuar. Mirpo nuk është e përshtatshme nësë ka ngarkesë të lartë e cila e ngadalson procesin. *Shpërndarë/distributuar* – Përdor më shumë se një databaz, zakonisht të ndara sipas të dhënave që ruajn dhe shërbimeve të cilat e përdorin data bazën. Janë të përshtatshme dhe të shpejta edhe nëse ka ngarkes të lartë. Mirpo duhet të sigurohet sinkronizim i të dhënave.

*Interface ( API ) të sistemit* – mundësojnë komunkimin me sistemet e tjera, përpunojnë inputet, ruajn të dhëna nga sistemet e tjera, lexojn të dhëna nga sistemet tjera.

Dizajnimi i kontrolli të qasjes dhe sigurisë përfshin dizajnimin për kontrollin : E *ndërfaqes së përdoruesit* – Autorizimet e përdoruesit, *Në nivelien e aplikacionit* – Transaksionet të jenë atomike. *Në Bazën e të dhënave* – për ndonjë anomali të të dhënave. *Në rrjetë* – Firewalls, qasjet.

*Dizajnimi modular i aplikacionit* ka të bëj me ndarjen e funksionaliteteve në pjesë të vogla të quajtura *module*. Secili modul duhet të jet i lehtë për tu modifikuar, fshirë, zëvendësuar, testuar në mënyr të izoluar ashtu që mos të ndikojn në pjesën e madhe të sistemit. *Moduli* është grup i funksioneve të grupuara, të cilat janë të ekspuzuara me *interface* duke i mundësuar moduleve tjera të komunikojn me njëra tjetrën. Karakteristik e moduleve është ripodorshmëria e cila e bën dizajnin modular më të dobishëm dhe të qëndrueshëm.

Procesi i zbërthimit të aplikacionit në module të pa varura quhet *modulizëm*.

Për të arritur dizajn modular efektiv ndikon një faktorë i rëndësishëm i quajtur *Pavartësia Funksionale* – që thotë se funksionet duhet të jen atomike të cilat kryejn një detyr të vetme duke ndërvepruar pak ose fare modulet e tjera.

Ky faktor mund të matet me dy kritere : *Cohesion* – tregon se sa elementet brenda një moduli ndërveprojn për të kryer një punë të përbashkët. *Coupling* – tregon se sa modulet janë të varura nga modulet tjera ose ndërvepron me modulet tjera. Dizajn i mirë quhet kur ka *high cohesion* dhe *low coupling*.

