Software Engineering

Introduction to Software Engineering

Software Engineering është fusha që mirret me përmirësimin e proceseve për të krijuar sisteme më kualitative dhe më pak të kushtueshme, sisteme të cilat do të jenë në përgjithësi më performant dhe më të saktë.

Klasifikimet e produkteve softuerike:

Produkte gjenerike - do të thotë një produkt që realizohet njëherë dhe i njejti produkt përdoret nga shumë përdorues. Shembull i produktetve gjenerike është Microsoft Office.

Produkte custom - do të thotë një produkt që realizohet vetëm për një masë të caktuar të përdoruesëve. Shembull i produkteve custom është MySEEU, pra produkt i cili përdoret vetëm nga persona të cilët i takojnë universitetit SEEU.

Dallimet esenciale gjatë zhvillimit të produkteve softuerike:
Në rastin e një bug, në produkte gjenerike, miliona përdorues të atij produkti do të preken ndërsa në produkte custom më pak përdorues do të preken.
Në aspekt të kërkesave, është shumë më e vështirë menaxhimi i produkteve gjenerike për arsye të kërkesave të shumta të përdoruesve, ndërsa në produktet custom mund të vendoset dhe menaxhohet më lehtë.

Atributet eseciale të një sistemi softueri:

Mirëmbajtja - ka kuptimin se sa lehtë mund të mirëmbahet një sistem softuerik

Varësia dhe siguria - ka kuptimin se sa mund të mvaremi dhe sa të sigurtë jemi kur e përdorim një sistem softuerik

Efikasiteti - ka kuptimin se sa shpejtë punon një sistem softuerik

Pranueshmëria - ka kuptimin nëse përdoruesit do ta pranojnë një sistem të caktuar softuerik

Aktivitetet e proceseve softuerike:

Specifikimi i softuerit - pjesa në të cilën e kuptojmë se çfarë duhet të bëjë sistemi.

Zhvillimi i softuerit - pjesa e dizajnit dhe zhvillimit të të shkruarit të kodit.

Validimi i softuerit - pjesa e aktiviteteve të cilat përfshijnë testimin e sistemit. Në validimin e softuerit përfshihen të gjitha mekanizmat me të cilat sigurohemi se një sistem është zhvilluar në bazë të kërkesave të caktuara.

Evolimi i softuerit - pjesa e aktiviteteve të lartëpërmendura të cilat në aspekt kohor ndodhin pas dorëzimit të sistemit.

Faktorët që ndikojnë në sistemet softuerike:

Heterogjeniteti - ka kuptimin se çdo sistem që e realizojmë, ndryshon prej një sistemi tjetër.

Biznesi dhe ndryshimi social - ka kuptimin se çdo softuer krijohet për suport të një biznesi tjetër. Në parim, çdo softuer krijohet për suport të produkteve të një biznesi i cili i takon një tjetër industrie.

Siguria dhe besimi - ka kuptimin se gjdo sistem tashmë komunikon përmes internetit dhe në këto raste të komunikimit shfaqen edhe probleme të sigurisë dhe besimit.

Shkalla - ka kuptimin se çdo sistem që e realizojmë duhet të jetë i zhvilluar në sisteme të shkallëve duke filluar nga sisteme embedded e deri tek sisteme Internet-based.

Llojet e aplikacioneve:

Stand-alone applications - sisteme të cilat ekzekutohen në një kompjuter apo në një procesor.

Interactive transaction-based applications - sisteme të cilat i takojnë më shumë grupit të web aplikacioneve. Shembull është E-banking.

Embedded control systems - sisteme tek të cilat hardueri dhe softueri shiten si një tërësi.

Batch processing systems - sisteme të cilat në background procesojnë të dhëna dhe nuk kanë pothuajse aspak interface.

Entertainment systems - sisteme të cilat përdoren për argëtime personale. Shembull është IPTV.

Systems for modeling and simulation - sisteme të cilat përdoren për modelim 3D të objekteve. Shembull është Archicad.

Data collection systems - sisteme të cilat vazhdimisht mbledhin të dhëna. Çdo sistem i cili ekziston sot, në një mënyrë të caktuar kë një sistem të tillë në vete.

Systems of systems - sisteme të cilat përdorin sisteme të jashtme për arsye se është vështirë të realizohen sisteme komplekse në vete dhe në rastet kur realizohet një sistem i tillë, zakonisht ndahet në servise të veçanta.

Rëndësia e web software engineering është aspekti se shumica e sisteme të cilat realizohen sot bëhen në information systems dhe të gjitha sistemet në infomation systems realizohen si web aplikacione. Arsyeja tjetër është se web-i ka ndikuar shumë në mënyrën e realizimit të softuerit duke filluar nga metodologjitë siç janë ato Incremental dhe Agile. Një aspekt tjetër është ripërdorimi i softuerit apo ndryshe edhe ripërdorimi i kodit përmes platformave si GitHub.

Etika e software enginnering

Konfidencialiteti - ka kuptimin se çdo njohuri të cilën e marim dhe çdo punë të cilën e krijojmë në kuadër të një kompanie, i takon punëdhënësit. Konfidencialiteti ka të bëjë me atë se nuk mund të përdorim resurse të një kompanie për një qëllim tjetër pa lejen e punëdhënësit ose lejes eksplicite.

Kompetenca - ka kuptimin se për diqka për të cilën nuk kemi njohuri, më mirë është të stepemi sesa të ndërhyjmë. Shembull në raste të sigurisë, në qoftë se nuk kemi njohuri, është më mirë që ta lëmë në dorë të dikujt tjetër i cili ka njohuri për atë fushë sesa të ndërhyjmë vetë.

Të drejtat e pronësisë intelektuale - ka kuptimin dhe ndërlidhet me konfidencialitetin ku gjdo dokumentim i kodit i cili krijohet për një kompani, i takon punëdhënësit të asaj kompanie.

Keqpërdorimi i kompjuterit - ka kuptimin se profesionistë të softuerit nuk duhet të përdorin aftësitë e tyre teknike për keqpërdorim të kompjuterëve të njerëzve të tjerë.

Software Processes

Copying with change është mënyra se si ndryshimet e sistemit ndikojnë në proces për arsye se çdoherë gjatë realizimit të një sistemi, do të ndodhin ndryshime dhe me këtë rast do të ketë edhe ndryshime të kërkesave.
Është me rëndësi që procesi të mund të përballojë ndryshime për arsye se çdoherë do të ketë ndryshime që do intrigohen prej një pale të tretë dhe çdoherë duhet ta ndjekim zhvillimin dhe efikasitetin e një procesi me qëllim që të përmirësohet.

Çfarë është një proces?
Procesi është kryesisht një grup aktivitetesh. Pra të gjitha aktivitetet që duhet të zhvillohen për ta krijuar një sistem, që mund të jenë, analizimi i kërkesave, krijimi i një modeli, definimi i arkitekturës së dizajnit, të shkruarit dhe testimi i kodit dhe shumë aktivitete të tjera. Kur kemi grup aktivitetesh të strukturura (Structured Set of Activities) do të thotë se kemi një radhitje në kryerjen e këtyre aktiviteteve dhe nuk mund të testojmë sistemin para se të shënojmë kodin, do të thotë se aktivitetet të rradhiten dhe të kenë lidhshmëri njëra me tjetrën.

Aktivitetet e lartëpërmendura mund ti kategorizojmë në 4 kategori:

Specifikimi (Specification) - tregon se çka duhet të bëjë sistemi. Pra çdo herë që shkruajmë detaje se çfarë do të bëjë sistemi atëherë lidhet me kategorinë e specifikimit.

Dizajni (Design) - tregon mënyren se si do të bëhet dhe zhvillohet sistemi. Kategoria e dizajnit kryesisht parashtron pyetjen how.

Implementimi (Implementation) - tregon se a është bërë ajo që duhet bërë. Kategoria e implementimit kryesisht parashtron pyetjen if.

Verifikimi (Verification) - tregon se a jemi korrekt dhe kompatibil me kërkesat që janë parashtruar në kategorinë e specifkimit.

Validimi (Validation) - në fund kur sistemi dorëzohet, analizohet edhe njëherë në detaje për arsye se mund të ndodhë që klienti ka kërkuar diçka tjetër dhe gjatë kësaj faze sigurohemi që ky sistem i përgjigjet kërkesave të klientit.

Evolimi (Evolution) – tregon se mund të ketë aktivitete nga specifikimi dhe testimi në mënyrë që të ndryshohet sistemi në bazë të kërkesave të klientit.

Procesi është mënyra se si i organizojmë aktivitetet. Për organizimin e aktiviteteve, na duhen disa informacione për secilin aktivitet në mënyrë që të mund të bëjmë një plan për zhvillimin e tyre.

Aspekti i parë i rëndësishëm është varësia (dependence) dhe nënkupton të kuptuarit nëse testimi varet prej programimit, sepse për të bërë testim nevojitet kod dhe gjithashtu duhet të dihet edhe produkti pasi ajo do të përcaktojë edhe lidhshmërinë se cili aktivitet do të kryhet para cilit aktivitet tjetër.

Aspekti i dytë i rëndësishëm janë rolet (roles) dhe tregon se kush duhet ta kryej një aktivitet të caktuar ose çfarë resursesh nevojiten për ta kryer një aktivitet. Në software engineering, resurset janë njerëz, por resurset mund të jenë edhe pajisjet si hardueri për të cilin ndoshta nevojitet rezervim për ta përdorur.

Aspekti i tretë i rëndësishëm është para dhe pas kushti (pre and post condition) ku tregohen mënyrat ekspilicite të kushteve, cilat parakushte duhet të plotësohen që të mund të fillojë një aktivitet, pra çfarë parakushte do të ketë në fillim që të dimë se çka do të dorëzohet në përfundim të atij aktiviteti, ose si do të jetë gjendja e sistemit pasi të ketë mbaruar ai akitvitet.

Mënyra se si organizohen detyrat (tasks) në aspekt kohor kryesisht është bërë në menaxhimin plan driven, pra prej fillimi është bërë një plan i detajuar dhe deri në fund mundohesh ti respektosh afatet e caktuara, gjë që është vështirë të realizohet dhe njëkohësisht sjell edhe probleme prandaj si shkak i kësaj erdhi forma e menaxhimit agile ku në fillim planifikojmë aq sa dimë dhe pastaj gradualisht vendosim detaje. Pra të gjitha aktivitetet që i kemi në menaxhimin plan driven do i përsërisim në kohë më të shkurtër duke shtuar në mënyre inkrementale aktivitete që do na nevojiten në ndërkohë.

Modeli Waterfall ose Plan-driven
Tek faza e parë është dizajni që nënkupton se si do të organizohet arkitektuara dhe në sa module. Dizajni në nivel më të ulët përfshin metodat, klasat, integrimin dhe testimin. Unit testing ka për qëllim që për çdo klasë të bëhet nga një testim për arsye se duhet të arrijmë në konkluzionin nëse funksionon apo jo dhe nëse ka probleme atëherë duhet të kthehemi prapa. Një nga mangësitë e Waterfall është që mund të hasim gabime gjatë këtyre fazave dhe që në përfundim mund të mos pëlqehet nga klienti.

Modeli Agile ose Incremental
Prej në fillim shkruajmë diqka të vogël, fillojmë me një version inicial që do ja japim klientit me qëllim që klienti të ketë feedback dhe në këtë mënyrë është më lehtë të bëhen ndryshime për aryse se marim kërkesa të reja, bëjmë specifikimin, zhvillimin, shkruajmë kod, bëjmë testimin dhe në fund do të dorëzohet te klienti. Përparësitë janë që mund të bëjmë ndryshime shumë më lehtë për arsye se do të kemi feedback nga klienti që në fazat e para dhe çdo 2 javë do mund të konsulltohemi për përmirësimin e atij sistemi, dhe çdo kërkesë që vjen e dimë saktë se çfarë duhet bërë me të, dhe klienti shumë shpejtë mund ta përdorë sistemin. Një nga mangësitë është se çmimi i sistemit do të caktohet në fund, pra nuk mund të përcaktojmë një çmim gjatë fazave fillestare. Gjithashtu, në Agile nuk mund të përgjigjemi saktë se deri ku kemi arritë, gjë që në Waterfall dihet më lehtë, si dhe është më i vështirë planifikimi i resurseve. Aspekti tjetër i rëndësishëm është që të bëhen ndryshime në mënyrë minimaliste, pra shtojmë kod në klasa që vetëm se ekzistojnë, dhe në Agile gjatë gjithë kohës është e rëndësishme funksionaliteti refunctioning, pra kodin ekzistues ta riorganizojmë në vazhdimësi.

Integrimi dhe Konfigurimi (Integration and Configuration) - ka në disa projekte ku nuk kërkohet të shkruhet shumë kod dhe vetëm bëjmë aktivitete tjera. Në këtë rast kemi më shumë procese të cilat tregojnë qartazi aktivitetet që ndodhin.
Avantazhet dhe disavantazhet – kur përdorim gjëra të gatshme, koha e problemit reduktohet shumë. Problemi më i rëndësishëm është që nuk e dimë se çfarë do të ndodhë pas 1 viti. Shembull në rast të ndërrimit të softuerit, do të duhet të ndryshohet komplet kodi.

Në aspekt të dizajnit kryesishtë janë disa lloje dizajnesh që duhet ti ndryshojmë në mënyrë që dizajne specifike të punojnë në çdo aplikacion. Organizimi i kodit, interface dizajni, UI dizajni, dhe API dizajni janë disa nga dizajnet të cilat duhet të ndryshohen për të arritur përdorimin e duhur.

Component design - nënkupton dizajn më në detaje. Kur bëjmë një arkitekturë atëherë realizimi i komponeneteve do të bëhet në disa raste si package dhe në disa raste të tjera si modul.

Database design – pasi realizohet arkitektura duhet edhe databaza të pësojë ndryshime.

Platform information - cilat teknologji janë përdorur.

Data description - shumë e rëndësishme për database design, se çfarë fusha përdor interface-i dhe çfarë fusha jo. Si output i kësaj faze është arkitektura e sistemit.

Implementation – të shkruhet kod dhe të bëhet unit testing për atë kod.

Validation - nënkupton verifikimin e pjesës së kodit dhe realizimit të unit testing ku për një pjesë të vogël e quajmë integration test dhe kur kompletohet i gjithë sistemi, atëherë thirret system test.

Llojet e testimeve:

User testing - testimi i realizuar nga useri.

Acceptance testing - testimi i realizuar nga klienti në mënyrë që të vendosë nëse sistemi pranohet apo jo.

Load testing - testimi i realizuar për rastet nëse sistemi pranon numër të madh të kërkesave.

Një mënyrë tjetër për ti zvogëluar ndryshimet janë prototipet. Në rastet kur realizon diqka shpejtë për ti treguar klientit, krijon një interface apo diçka që është e dukshme.

Configuration Management

Checksum is a simple hash code.

Git commands:

git init - creates an empty Git repository or reinitialize an existing one.

git status - displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

git add - moves changes from the working directory to the staging area. This gives you the opportunity to prepare a snapshot before committing it to the official history.

git commit -m "" - Takes the staged snapshot and commits it to the project history.

git log - lets you explore the previous revisions of a project. It provides several formatting options for displaying committed snapshots.

git diff - lists out the changes between your current working directory and your staging area.

git branch - lets you list, create, or delete branches.

git checkout - in addition to checking out old commits and old file revisions, git checkout is also the means to navigate existing branches.

git merge - a powerful way to integrate changes from different branches. After forking the project history with git branch, git merge lets you put it back together again.