View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Business and Technology in Kosovo: UBT Knowledge Center Collections

University of Business and Technology in Kosovo

UBT Knowledge Center

Theses and Dissertations

Student Work

Spring 3-2011

Advances in Software Engineering

Alban Krasniqi

Follow this and additional works at: https://knowledgecenter.ubt-uni.net/etd

 Part of the Computer Sciences Commons

Universiteti për Biznes dhe Teknologji (UBT)

Advances in Software Engineering

PUNIM DIPLOME

Alban Krasniqi

Mars / 2011
Prishtinë

Universiteti për Biznes dhe Teknologji (UBT)

Punim Diplome
Viti Akademik 2010-2011

Alban Krasniqi

Advances in Software Engineering

Profesor Dr. Astrit Hyseni

14 / 03 / 2011

ABSTRAKT

Inxhinieria Softuerike është aplikimi i një qasje sistematike të disiplinuar, të matshme për zhvillimin, operimin dhe mirëmbajtjen e softuerit.

Për rreth çdo aktivitet është një strukturë komplekse të cilat janë të mbështetura dhe të lidhura me aktivitetet dhe shqetësimet ekonomike, organizative, menaxheriale dhe një strukture komplekse të kufizuar.

Ne këtë punim jemi munduar qe ti trajtojmë disa tema qe i gjejmë të trajtuara rrallë neper literaturat qe ofrohen sot ne fushën e inxhinierisë softuerike .

Duke pas parasysh se Inxhinieria Softuerike përfshin një diapazon të gjere mirëpo nder pozitat me kryesore është ajo e menaxhimit te projekteve ne me ane te këtij punim jemi munduar qe te prezantojmë sa me shumë informacione për problemet, arritjet dhe pritjet e për te ardhmen si dhe aplikimin e disa standardeve të përgjithshme .

PËRMBAJTJA

1. HYRJE

Është Termi "Inxhinieri Softuerike" hyri në përdorim të përbashkët si rezultat i Konferencës së parë të NATO-s në Software Engineering me 1968 . Komiteti i Shkencës të NATO-s kishte zgjedhur fraza sepse ajo sugjeroi nevojën për ndërtimin e softuerëve të cilët duhet të jene në bazë të llojeve të themeleve teorike dhe disiplinave praktike që janë tradicionale në degët e inxhinierisë së themeluar . Një krizë softuerike e projekteve të dështuara dhe sistemeve softuerike të pakënaqshme ishin të njohura gjerësisht dhe shumë të përfolura . Në anën tjetër inxhinierët që projektuan dhe ndërtuan automobila, apo aeroplanë ose ura duket se kanë arritur nivele shumë më të lartë të suksesit dhe besueshmëri.

Pamja e komitetit ishte e qartë : zhvilluesit e softuerëve duhet të mësojnë për ti respektuar dhe inxhinierët që kanë miratuar teori të ngjashme apo praktika, qofshin të njohura tashmë apo ende të pa hartuara.

1.1 CFAR ËSHTË INXHINJERIA SOFTUERIKE ?

Ashtu si inxhinierë të automobilave që zhvillojnë automobila, edhe inxhinierët e softuerëve zhvillojnë softuer. Në të dy rastet, inxhinieri synon të zgjidhë ndonjë problem, për të kënaqur disa nevoja të njeriut.

Për rreth çdo aktivitet është një strukturë komplekse të cilat janë të mbështetura dhe të lidhura me aktivitetet dhe shqetësimet ekonomike, organizative, menaxheriale dhe një strukturë komplekse të kufizuara në mënyrë të barabartë .

1.2 NATYRA E NDRYSHIMIT TË SOFTUERVE

Softueri është bërë pjesë e integruar në shumicën e fushave të jetës njerëzore.

Sistemet softuerike

Infrastruktura softuerike ka ardhur në këtë kategori si kompajleret, sistemet operative etj. Në parim sistemet softuerike janë një koleksion i programeve që ofrojnë shërbime për programe të tjera.

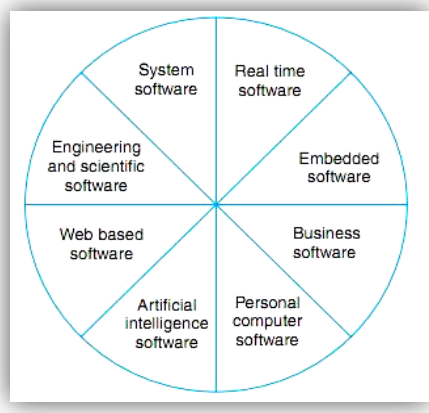

Figura 1 Natyra e ndryshimit të softuerve

Real time softuerët :

Këto programeve janë përdorur për të vëzhguar, kontrolluar dhe analizuar ngjarjet në botën reale pasi ato të ndodhin. Një shembull mund të jetë softuerët e nevojshëm për parashikimit të motit .

Softuerët e tillë do të mbledhin dhe përpunojnë të dhënat e statusit të temperaturës, lagështisë dhe parametrave të tjerë në mjedis për të parashikuar motin .

Embedded softuerët

Ky lloj i softuerit është vendosur në memorien vetëm për lexim (ROM Read Only Memory) të produktit dhe kontrollojnë funksioneve të ndryshme të produktit. Produktit mund të jetë një avion, automobil, sistemi i sigurisë, sistemi i sinjalizimit, etj Softuerët Embedded merren me komponentët harduerike dhe është quajtur edhe si softuer inteligjent .

Softuerët e Biznesit

Përfshin një zone të madhe të zbatimit . Këta softuerë janë dizajnuar për procese të biznes aplikacioneve andaj edhe quhen biznes softuer. Biznes softuer mund të jetë një komponent ku ndihmon për të marrë vendime bazuar në të dhënat në dispozicion . Sistem për menaxhimin e informacioneve (MIS), ERP dhe programe të tilla të tjera janë shembuj të njohura të programeve të biznesit.

Softuerët e Kompjuterëve Personal:

Softuerët që përdoren në kompjuterët personal janë të përfshirë në këtë kategori. P,sh . Programe për procesim të tekstit, grafikes, multimedia dhe animacione të ndryshme, menaxhimi i bazës së shënimeve, lojërat kompjuterike etj.

Kjo është një fushë ku shumë organizata të mëdha i kanë përqendruar përpjekjet e tyre këtu për shkak të numrit ë madhe të konsumatorëve.

Softuerët e Inteligjencës Artificiale :

Softuerët e Inteligjencës Artificiale përdorën në fushën algoritmeve jo-numerike për të zgjidhur problemet komplekse të cilët nuk janë të përshtatshëm për llogaritje

Shembuj , sistemet eksperte, rrjetat artificiale neurale, softuerët për procesim të sinjalit etj

Ueb softuerët:

Softuerët që lidhen me ueb aplikacione si : CGI, HTML, Java, Perl, DHTML etj.

Softuerët Shkencor dhe Inxhinierik

Softuerët shkencor dhe ata inxhinierik janë grupuar në këtë kategori. Nevojiten kompjuter të mëdhenj për procesim të dhënave . Shembuj : CAD/CAM package, SPSS, MATLAB, Engineering Pro, Circuit analyzers etj.

2. KËRKESAT E SOFTUERIT, ANALIZA E BIZNESIT DHE ARKITEKTURA

Para çdo kodimi në një aplikacione softuerik është e nevojshme për të përcaktuar karakteristikat, strukturën, fushëveprimin, dhe user interfaces qe do të zhvillohen

Është gjithashtu e nevojshme për të përcaktuar metodat e ofrimit të këtyre tipareve dhe platformën në të cilën do të veprojë. Përveç këtyre objektiva dhe synime të tjera për aplikacione duhet të përcaktohen në bazë të përformances, sigurisë , besueshmërinë dhe një numër i temave të tjera.

Këto çështje të ndryshme janë përhapur në mesin e një numri të dokumenteve dhe planeve që përfshijnë kërkesat, analizën e biznesit, arkitekturën dhe dizajnin .

Edhe pse një numër i templateve dhe modeleve ekzistojnë për çdo lloj dokumenti, nuk ka asnjë metodë që ka provuar të jetë tërësisht e suksesshëm. Edhe pas më shumë se 60 vite në zhvillim të softuerëve, një numër i problemeve të përbashkëta ende ndodhin pothuajse të gjitha aplikacionet e mëdha softuerike .

Problemet janë tipike për industrinë e softuerit :

1. Kërkesat rriten dhe ndryshojë me ritme më të madhe se 1 për qind për çdo muaj kalendarik .
2. Aplikacionet përfshijnë më pak se 80 për qind të kërkesave të përdoruesit në versionin e parë.
3. Disa kërkesa janë të rrezikshme ose "helmuese" dhe nuk duhet të përfshihen.
4. Disa aplikacione janë të mbushura shumë me karakteristika qe askush nuk i kërkon.
5. Shumica e aplikacioneve softuerike janë të mbushura me dobësi të sigurisë.
6. Gabimet në kërkesa dhe dizajnim do të shkaktojë shumë ashpërsi të lartë të defekteve.
7. Metoda efektive të tilla si inspektimi i kërkesave dhe dizajnimit janë përdorur rrallë.
8. Standardet , e kërkesave dhe dizajnit të riperdorueshëm nuk janë shumë të vlefshme
9. Dokumentimet shumë komplekse për t'i kuptuar njerëzit .

2.1 KËRKESAT SOFTUERIKE

Inxhinierët e softuerit kanë përgjegjësi për të ndihmuar klientët të përcaktojnë kërkesat në mënyrë të plotë dhe efektive

Kjo është punë e inxhinierëve softuerik profesional të cilët këmbëngulin në kërkesat efektive të metodave të projektimit të tilla si : joint application design (JAD), quality function deployment (QFD) dhe requirements inspections .

Është gjithashtu përgjegjësi e inxhinierëve të softuerit për të njoftuar klientët për të gjitha kërkesat që potencialisht janë të dëmshme.

Shumë shpesh në literaturë hasim në kërkesat e softuerike dhe i shohim disa supozime të gabuara që përdoruesit do të duhej jetë 100 për qind efikas në identifikimin e kërkesave. Ky është një supozim i rrezikshëm. Kërkesat e klientit kurrë nuk janë komplete ato shpesh janë të gabuara. Ajo duhet të jetë përgjegjësi e inxhinierëve të softuerit të cilët duhet të insistojnë që të përdorin metoda si : gjurmimin e të dhënave për aplikacionet e trashëguara , JAD, QFD, prototipet , kërkesat e inspektimit , Use Case po ashtu janë të rekomanduara

2.1.1 Struktura dhe përmbajtja e kërkesave softuerike

Kërkesat softuerike qartë përshkruajnë karakteristikat kryesore dhe funksionet që një aplikacion softuerik do të përmbajë. Por kërkesat e specifikimeve do të duhej të shërbejnë edhe për qëllime të tjera të biznesit.

Për shembull, kërkesat duhet të diskutojë çdo limit apo kufizim të softuerit, të tilla si kriteret e përformances, kriteret e besueshmërisë,kriteret e sigurisë dhe te ngjashme. Kostos dhe skegjullimi i ndërtimit të aplikacioneve softuerik janë të ndikuara fuqishëm nga madhësia e aplikacionit në drejtim për të përcaktuar kërkesat e përgjithshme që do të zbatohen. Prandaj kërkesat janë baza kryesore për të sqaruar madhësinë e softuerit .
Në mënyrë kronologjike, këto tema themelore duhet të hulumtohen si pjesë e kërkesave të procesit të mbledhjes:

1. Rezultatet që duhet të jetë prodhuar nga aplikacionet.

2. Inputet që do të hyjë në aplikacionin softuerik.

3. Dosjet logjike që duhet të ruhen ne aplikacione.

4. Entitetet dhe relacionet qe do të jene fajlla logjik të aplikacionin.

5. Lloje informacioneve të dhënash që mund të përdoren për aplikacione.

6. Interfejsat në mes të aplikacionit dhe sistemet e tjera.

7. Algoritme kryesore të cilat duhet të jenë të pranishëm në aplikacione.

2.1.2 Kërkesat softuerike metodat dhe praktikat

Ka dallime të shumta në mënyrën se si kërkesat softuerike janë mbledhur, analizuar, dhe konvertuar në softuer. Disa nga ato janë diskutuar sipas rend alfabetik.

Fokus grupet

Një fokus grup është një asamble e konsumatorëve të cilët janë kërkuar për të marrë pjesë në diskutimet në grup në lidhje me karakteristikat dhe funksionet e produkteve të reja.

Fokus grupi zakonisht varion nga 5 në ndoshta më shumë se 25 pjesëmarrës në nevojat bazuar demografike të potencialit të produktit . Fokus grupet mund të ofrojnë sugjerimet ne lidhje me përdorimin e modeleve të punës dhe prototipeve. Fokus grupet kanë provuar të jenë efektive për produktet që kanë për qëllim një përzierje të interesave të ndryshme dhe shumë llojshmëri të përdorimit. Fokus grupet janë më të vjetër se softuerët dhe janë përdorur shpesh për pajisje elektronike, pajisje, dhe objekte të tjera të prodhuara . Në një kontekst softuerik, fokusgrupet janë më të efektshme për aplikacionet softuerike komerciale që synon qindra ose mijëra përdoruesve, ku diversiteti është pjesë e qëllimeve të aplikacionit .

Joint application design (JAD)

Koncepti i projektimit në IBM Toronto si një metodë për mbledhjen e kërkesave për aplikacionet financiare. Metoda normale e kryerjes JAD është për një grup të palëve të interesuara ose të përdoruesit të takohen ballë për ballë me një grup të arkitektëve softuerik dhe projektues në një mjedis formal me moderator.

Shpesh takime JAD zhvillohen në vende jashtë objekteve. Ndërmjet tre dhe dhjetë përdor-ues takohen me një grup prej tre deri në dhjetë arkitektët apo dizajner softueri . Takimet e zgjasin nga zakonisht 2 ditë në më shumë se 15 ditë, në bazë të madhësisë së aplikacionit për diskutim.

Unified modeling language (UML)

Gjuha e modelimit UML është një pjesë integrale e procesit të unifikuar racional (Rational Unified Process (RUP)) që është në pronësi tani nga IBM.

Historia e uml është një bashkim i koncepteve të Grady Booch, James Rumbaugh, dhe Ivar Jacobsen që të njohur edhe në mesin e komunitetit softuerit . Uml dhe paraardhësit e tij kishin për qëllim në fillim tek mbështetjen e kërkesave të object-oriented dhe dizajnimin, por në fakt tani mund të mbështesë pothuajse çdo formë të softuerit.

UML është një grup i pasur dhe kompleks i notacioneve grafike që përfshijnë jo vetëm kërkesat por edhe arkitekturën , dizajnimin e bazës së të dhënave, dhe objekte të tjera të aplikacionit . Për kërkesat dhe karakteristikat e riperdorueshme që do të përdoren nga aplikacione të shumta, kjo do të ishte e dobishme që të ketë një lloj agjenti intelegjent me bashkim te modeleve që mund të pastroj diagramet uml dhe nxjerrjen e modele të ngjashme.

UML nuk është zgjidhja, por Management Object Group (OMG) është vazhdimisht duke punuar për të shtuar karakteristika të dobishme dhe për të eliminuar elemente të mundim-shme . Prandaj, uml ka të ngjarë që të zgjerohet në të ardhmen .

Në teori, është e mundur për të zhvilluar një komponent që do të krijoj automatikisht funksione nga parsimi i diagrameve të ndryshme UML. Në fakt, mjetet e tilla eksperimentale janë ndërtuar .

2.2 BIZNES ANALIZA

Fraza "biznes analizë" është shumë e ngjashme me frazën e vjetër "sistemet e analizës."

Shumë korporata punësojnë specialistë të analizave të biznesit që të shërbejë si një ndërlidhës ndërmjet komunitetit të inxhinierisë softuerike dhe njësive operative të kompanisë. Për shkak të rolit të tyre si ndërlidhës në mes të komuniteteve teknike dhe të biznesit, Analistët e biznesit janë të përfshirë shumë herët dhe janë pjesëmarrësit kryesorë edhe para së të fillon nxjerrja e kërkesave .Analistët Biznes vazhdojnë të jenë të përfshira gjatë dizajnimit (projektimit) dhe pjesën e hershme të fazës së kodimit , për të analizuar rrjedhjen dhe bashkëngjitjen e kërkesave deri edhe në fazën e kodimit .

Rolet e analistëve të biznesit janë për të ndihmuar në nxjerrje e kërkesat , dhe për të siguruar që të dyja anët ajo e teknologjisë së informacionit dhe anën e konsumatorëve ose palëve të interesuara të komunikojnë në mënyrë të qartë dhe në mënyrë efektive .Pozita e analistëve biznesit shumë është populluar prej sistem analisteve, inxhinierëve të softuerit, apo specialistë të sigurimit të cilësisë qe dëshirojnë qe ti zgjerojnë përgjegjësit e tyre. Biznes Analistët duhet të kenë një prapavijë në të dy fushat në fushën e softuer dhe ne atë të biznes dhe ata duhet të jenë ne gjendje që të lehtësojnë nxjerrjen e kërkesave dhe t'i analizojnë ato . Për shembull, analistët e biznesit janë shpesh moderatorët ne JAD (joint application design) seancave. Analistët e biznesit mund të marrin pjesë në inspektime të kërkesë, vendosjen e cilësisë së funksionit (quality function deployment - QFD), dhe aktivitete të tjera si në mbledhjen e kërkesat ose analizimin e tyre dhe shpjegimin e kuptimit të tyre për komunitetin e softuerit. Disa mangësi të dukshme në rolet e analistëve të biznesit ku shpesh kërkohen lloje të tjera të specialistëve. Për të ilustruar disa nga këto boshllëqe:

1. Madhësinë dhe vlerën e projektit softuerik .
2. Menaxhimin e fushëveprimi (objektivave) të projekteve softuerike .
3. Analizën e rrezikut të projekteve softuerike .
4. Ndjekja dhe monitorimi e progresit të projekteve softuerike .
5. Kontrollin e cilësisë të projekteve softuerike .
6. Analizën e sigurisë dhe mbrojtjen e projekteve softuerike.

Arsyeja për pohimin se këto zona paraqesin "boshllëk" është, sepse problemet janë shumë të zakonshme në të gjitha gjashtë fushat pavarësisht nëse analiza e biznesit është pjesë e procesit të kërkesave

2.3 ARKITEKTURA E SOFTUERIT

Në thelb të arkitekturë softuerike këto shtatë tema janë më themelore :

1. Struktura e përgjithshme e një aplikacione softuerik .
2. Struktura e përdorimit të dhënave ne aplikacione softuerik
3. Interfaces në mes të aplikacioneve softuerike dhe botës së jashtme
4. Zbërthimi e aplikacionit në komponentët funksionale
5. Lidhjet apo transmetimet e informacionit ndërmjet komponentëve funksionale
6. Përformanca e atributet që lidhen me strukturën
7. Atributet e sigurisë që lidhen me strukturën

Ka tema të tjera të lidhura, por këto shtatë duket të jetë tema themelore .

Rolet e arkitektëve softuerik dhe enterprise arkitektëve janë zhvilluar në vitet e fundit dhe do të vazhdojë të zhvillohet si tema të reja të tilla si cloud computing, service oriented architecture (SOA), dhe virtualization. Rëndësia e arkitekturës softuerike ngjan në rëndë-sinë e arkitekturës së shtëpive dhe ndërtesave. Nga rastësi, madhësia e një ndërtese fizike matet në terma të "metra katrore" dhe madhësinë e një aplikacioni softuerik matet në terma të "pikave të funksionit" modelet janë pjesë identike kur është fjala për rëndësinë apo vlerën e arkitekturës të mirë . Sistem shumë të madhe në varg të madhësinë, Orale, SAP ndoshta nuk do të jetë edhe e mundur pa arkitekturë shumë të mira dhe një numër specialistësh arkitektonike. Këto aplikacione kryejnë 100.000 pikë funksioni, ose më shumë se 5 milion deklarime në një gjuhë të tilla si Java (ndoshta më shumë se 15 milionë në fakt) . Arkitekturën softuerike dhe arkitektura e ndërtesave janë shqetësuar në masë të madhe me çështjet strukturore.

Megjithatë, arkitektura softuerike është edhe më e komplikuar ne ndërtimin e arkitekturës për shkak se aplikacionet softuerike nuk janë statike pasi qe ato të ndërtohen . Ato rriten vazhdimisht në rreth 8 për qind në vit, në shtimin e karakteristikave të reja . Kur operohet me aplikacione softuerike ato kanë një strukturë shumë dinamike të cilat mund të ndryshojnë me shpejtësi për shkak të thirrjeve dhe karakteristika të hapura dhe ato ndryshohen gjatë ekzekutimit. Prandaj, arkitektët e softuerit duhet të merren me çështjet dinamike dhe të përformances së punës .

Një tjetër ndryshim i rëndësishëm midis arkitekturës se ndërtimit dhe arkitekturë softuerike është në fushën e sigurisë. Sigurisht, për disa ndërtesa të tilla si Pentagoni dhe selia e CIA-s siguria është një shqetësim i lartë arkitektonik, por siguria nuk është zakonisht një çështje e madhe arkitektonike për shtëpitë e zakonshme dhe ndërtesat e vogla apo zyrat. Për aplikacione softuerike, kërkesa e sigurisë është duke u bërë gjithnjë e më e rëndësishme për të gjitha nivelet . Një arsye për këtë është për shkak se arkitekturat softuerike janë zhvilluar me shpejtësi. Kur aplikacionet ishin të vogla me mesatarisht më pak se 1000 pika të funksionit, ashtu siç u zhvilluan deri në fund të viteve 1960, arkitektura softuerike nuk ishte një temë me interes.

Edsger Dijkstra dhe David Parnasit diskutohet së pari arkitekturën softuerike si një temë me rëndësie rreth vitit 1968. Më vonë pionierë të tilla si Mary Shaw dhe David Garlan vazhduan të theksoj se arkitektura software ishte një faktor kritik për suksesin e sistemeve të mëdha.

Arsyeja për rëndësinë në rritje të arkitekturës ishte për shkak të katër faktorëve:

1. Aplikacionet softuerike ishin në rritje të shpejtë dhe kaluan 10.000 pika të funksionit ose 1 milion deklarime në kodin burimor . Sot madhësi mund të jenë dhjetë herë më të mëdha .
2. Volumi i të dhënave që përdorej nga aplikacionet e softuerike ka qenë në rritje edhe më shpejt se vetë softuerët . Numri i regjistrimeve të automatizuara të rritura nga mijëra në miliona e miliarda dhe vazhdon të rritet. Nuk ka dyshim trilionë të dhëna janë vetëm mbi horizont.
3. Bazat e të dhënave dhe të dhënat e skemave organizative janë zhvilluar shpejt apo më shpejt se skemat e arkitekturës softuerike.
4. Aplikacionet softuerike nuk u operoishin të gjitha të vetme në një kompjuter.

Kur aplikacionet e madhe softuerike filluan të jetë të ndara në komponentë që mund të veprojnë në mënyrë paralele, ose të veprojë në kompjuter të veçantë në të njëjtën kohë, arkitektura u bë një temë shumë e rëndësishme.

Aplikacionet softuerike u zhvillua vetëm për një kompjuter. Një nga nisje të rëndësishme nga ky model ishte që të ketë disa funksione të ekzekutimin në një kompjuter host (shpesh një mainframe), ndërsa funksionet e tjera të operuara në kompjuterët personal. Kjo metodë e shpërbërjes ishte quajtur arkitekturë clientserver.

Në vitet 1980 dhe edhe më shumë në vitet 1990, shumë qasje të tjera arkitektonike u shfaqen . Ata përfshihen, por nuk ishin të kufizuara në , event-driven architecture , three-tier architecture (shtresa e prezanitimit , shtresa e biznes logjikes, dhe shtresa e databazes), N-tier architecture, peer-to-peer architecture, model-driven arkitekturë, dhe natyrisht se arkitekturat më të fundit pattern-based architecture, service-oriented architecture (SOA)

Natyrisht, arkitektët softuerike duhet të marrin në konsideratë bashkimin në mes të strukturës së programeve në vetvete dhe të dhënave , optimizmin e të dhënave organizatave për të përmbushur qëllimin e aplikacionit. Këto nuk janë zgjedhje të parëndësishëm, përvoja dhe njohuri të veçanta janë të nevojshme. Për të pasur sukses në zgjedhjen dhe dizajnimin e aplikacioneve duke përdorur ndonjë nga këto forma më të fundit të programeve dhe të dhënave të arkitekturës u bë një punë që kërkon trajnime të veçanta dhe përvojë të konsiderueshme.

Si rezultat, shumë kompani të mëdha të tilla si IBM dhe Microsoft krijuar përshkrimet e të reja të punës dhe të titujve të të ri të punës të tilla si "arkitekt" dhe "senior arkitekti ." Kur filluan te shfaqen pozitat e arkitektit në kompanitë e mëdha , disa shoqatave u shfaqen në mënyrë që arkitektët të mund ndajnë informacion dhe të kenë qasje në të menduarit e fundit. Një nga këto është Shoqatës Ndërkombëtare e Arkitektëve të Softuerit (International Association of Software Architects - IASA), dhe një tjetër është World

Wide Instituti i Arkitektëve Softuerike (World Wide Institute of Software Architects - WWISA). Ka edhe revista të specializuara si Microsoft Architecture Journal që kanë të bëjnë me tema arkitektonike.

2.4 ENTERPRISE ARKITEKTURA

Nevoja për enterprise arkitekturë është rritur ne mënyrë progresive gjatë 30 viteve të fundit, për shkak se një pjesë e madhe e kompjuterëve dhe softuerëve u bënë pjesë e operimeve të korporatave . Në fund të viteve 1960, kur kompjuterët e parë mainframe filluan të përdoren për problemet e biznesit, aftësitë e tyre ishin disi primitive dhe të kufizuara. Si rezultat, aplikacione e hershme të biznesit priren të jenë shumë të lokalizuara , për të vepruar në një kompjuter të veçantë dhe në një qendër të dhënave të veçanta, për t'i shërbyer vetëm një numri të kufizuar të shfrytëzuesve në një njësi të vetme të biznesit .Korporatat kanë njësi të shumta që veprojnë, duke përfshirë prodhim, marketing , shitjen,financat, burimet njerëzore, si dhe të tjera .

Korporatat e mëdha gjithashtu kanë biznese të shumta dhe fabrika të shpërndara nëpër qytete të shumta . Kur kompjuterët dhe softuerët e parë u bën mjetet të biznesit, ajo ishte një praktikë e zakonshme ku për çdo njësi operative të kanë qendrën e saj të dhënave dhe të zhvillojnë softuerë specifik. Shpesh ka pasur pak ose aspak komunikim mes njësive operative, si në karakteristika , interface ose të dhënat të aplikacioneve që ishin të automatizuara .

Deri ne vitet 1980, korporatat e mëdha kishin zhvilluar qindra apo edhe mijëra aplikacioneve softuerike, ku shumica e të cilave kanë shërbyer vetëm për qëllime të ngushta dhe lokale . Kur zyrtarëve të korporatave i nevojiteshin informacione të konsoliduar nga të gjitha njësitë e biznesit, shpenzonin shumë kohe dhe ishte punë e shtrenjtë por e nevojshme për nxjerrjen e të dhënave nga kërkesat e ndryshme ku duhej të prodhohehsin raporte të konsoliduara. Kjo situatë e pakëndshme shkaktoi shfaqjen e arkitekturës enterprise si një disipline kryesore për të sjellë qëndrueshmëri të përpunimit të të dhënave në të gjithë njësitë operative. E njëjta situatë edhe shkaktoi shfaqjen në treg softuerëve të rëndësishëm komercial : enterprise resource planning (ERP) .

Koncepti bazë i aplikacioneve ERP është që aplikacionet individuale janë shumë të vështira për t'u lidhur së bashku , që kjo të jetë më e lirë për të zëvendësuar të gjithë ato me një sistem të vetme të madhe që mund t'i shërbejë të gjitha njësitë që veprojnë në të njëjtën kohë, dhe për të ruajtur të dhënat në një format të qëndrueshëm e që ka shërbyer korporatave dhe njësive . Dallimi kryesor në mes arkitekturës siç diskutuam në seksionin e mëparshëm dhe arkitekturës enterprise është fusha e përgjegjësisë.

Normalisht, arkitektët punojnë në një aplikacione individual, të cilat mund të shkojnë nga 10.000 në më shumë se 100.000 pikë funksioni. Arkitektët Enterprise punojnë në portofol-iot e korporatave të cilat mund të variojnë nga rreth 2 milionë pikë funksion të më shumë se 20 milion pika të funksionojnëve në madhësinë total.

Portofoliot e korporatës për kompanitë e mëdha të tilla si Microsoft, IBM, ose Lockheed përmbajnë mijëra aplikacioneve. Megjithatë, një tjetër aspekt i arkitekturës enterprise është fakti se korporatat e mëdha krijojnë dhe përdorin lloje të ndryshme të programeve: aplikacioneve konvencionale të teknologjisë së informacionit, ueb aplikacione, aplikacione embedded dhe sisteme softuerike .Arkitektura Enterprise ka të njëjtën ndërlidhje me arkitekturën e planifikimit urban që është një arkitekturë e ndërtimit. Me arkitekturën e ndërtimit, një arkitekt ka të bëjë kryesisht me një ndërtesë të vetme. Por, planifikuesit urban duhet të jenë të shqetësuar për mijëra ndërtesa në të njëjtën kohë. Planifikuesit urban duhet të mendojnë se çfarë lloj të infrastrukturës do të jetë e nevojshme për të mbështetur sektorë të ndryshëm të tilla si banimin, tregjet ,industritë , e kështu me radhë .

Disa shoqata jofitimprurëse mbështesin fushën enterprise arkitekturës. Disa nga këto janë : Shoqata e Arkitektëve Enterprise (Association of Enterprise Architects - AEA), Association of Open Group Enterprise Architects (AOGEA), kjo e fundit është formuar me bashkimin e dy shoqatave Open Group organization dhe Global Enterprise Architecture Organization (GEAO) . Ky grupi i shkrirë pohon se ajo është bërë organizata më e madhe e arkitektëve në botë. Kanë edhe një revistë për enterprise arkitekturë qe quhet : The Journal of Enterprise Architecture (JEA), . Është e vështirë të gjeni informacion në lidhje me planet specifike të arkitektëve enterprise për korporatat, sepse puna e tyre zakonisht është e pronarit-ve të korporatës dhe nuk vihen në dispozicion të publikut. Nuk ka dyshim se puna e arkitekturës enterprise do të vazhdojë të evoluojnë me ndryshime teknike dhe të biznesit .

3. MENAXHIMI I PROJEKTEVE DHE INXHINJERIA SOFTUERIKE

Menaxhimi i projektit është një lidhje e dobët me inxhinierin softuerike . Po ashtu është gjithashtu një lidhje të dobët në kurrikulat e arsimit akademik në shumë universiteteve. Shumë probleme softuerike dhe probleme të tilla si : kostot , tejkalimet e planifikuara mund t'i atribuohen menaxhimit të dobët të projektit se sa të programimit të dobët ose praktikave të dobëta të inxhinierisë softuerike . Përmirësimi i menaxhimit të projektit është në rrugën kritike për reduktimin e suksesshëm të kostos . Menaxhimi i projektit duhet të definohen në një kontekst softuerik . Termi i menaxhimin të projektit është ngushtuar artificialisht nga shitësit e komponentëve është bërë e kufizuar në aktivitetet e analizës rrugën kritike dhe prodhimin e ndihmave të ndryshme të tilla si planifikimi i gjallë dhe skemave të gantogramit .

Termi i menaxhimit të projektit është ngushtuar artificialisht nga shitësit komponenteve në mënyrë që ajo është bërë e kufizuar në aktivitetet e analizës rrugën kritike dhe prodhimin e ndihmave të ndryshme të tilla si planifikimi i PERT dhe Gant diagrameve . Vëzhgimet e bëra gjatë viteve të fundit tregojnë se menaxhimi i projektit është shumë nën mesataren në shumë aktivitete kritike. Arsyeja kryesore e përformances nga menaxherët e projekteve softuerike është ndoshta mungesa e programeve efektive në universitet dhe nivelit e shkollimit pas-universitarë . Pak inxhinier të softuerit dhe edhe më pak studentë MBA nuk kanë mësuar asgjë në lidhje me vlerën ekonomike të cilësisë së programeve apo si matet efikasiteti apo nivelet e largimit të defekteve, të cilat janë në të vërtetë të vetmet matje të rëndësishme në inxhinierin softuerike . Supozimin se një kurrikul shumë më e përmirësuar për menaxherët e projektit mund të vihet në dispozicion brenda dhjetë viteve, i shoqëruar me supozimin se menaxherët e projektit atëherë do të jetë e pajisur me matje moderne, duke llogaritur koston, duke vlerësuar cilësisë, dhe mjetet matëse.

Për fat të keq, teknologjia e projektit të menaxhimit të programeve është shumë më e mirë se përformanca aktuale e menaxherëve të projekteve. Në vitin 2009, vetëm rreth 5 për qind e projekteve softuerike në SHBA krijuar standardet e produktivitetit dhe cilësisë së të dhënave . Më pak se gjysma e 1 për qind kanë paraqitur pikat e referimit të të dhëna për zyrtaret e standard si : Software Benchmarking Standards Group (ISBSG), David Software Productivity Research (SPR), David Consulting Group ose organizatave të ngjashme. Çdo projekt i rëndësishëm softuerik duhet të përgatisë kriteret formale në përfundimin e projektit. Një gjeneratë e re e projektimit të integruar të menaxhimit të komponentëve softuerike po afrohet, e cila ka premtimin e eliminimit të mangësive në komponentët aktuale të menaxhimit të projektit dhe përmirësimin e aftësisë për të shkëmbyer informacione nga një komponent në tjetrin . Klasa e re e komponentëve

të menaxhimit të projekteve të tilla si mjetet e menaxhimit të metodologjisë janë bashkuar gjithashtu për të vendosur në dispozicion të komunitetit të menaxhimit të softuerëve . Menaxhimin e projektit softuerike është një nga punët më të kërkuar të shekullit të 21. Projekti menaxherët e softuerëve janë përgjegjës për ndërtimin e disa prej aseteve më të shtrenjta që korporatat kanë tentuar ndonjëherë për të ndërtuar. Për shembull, të sistemet e mëdha softuerike kanë kosto shumë më të madhe për ti ndërtuar se sa ndërtimi shumë ndërtesave apo zyrave të kompanisë . Më të vërtet sistemet e mëdha softuerike me 100000-funksione mund të kushtojë më shumë se ndërtimi i një stadiumi të futbollit një rrokaqielli 50 katesh ,apo një anije 70.000 ton që lundron në det. Jo vetëm që sistemet e mëdha të softuerëve janë të shtrenjta, por ata gjithashtu kanë një nga normat e dështimit më të lartë të ndonjë objekti të prodhuar në historinë njerëzore . Termi dështim i referohet projekte që janë anuluar pa përfunduar për shkak të kostove ose tejkalimit te planifikimeve Për dështimet dhe fatkeqësitë softuerike , pjesa më e madhe e fajit mund ti bie për komunitetin e menaxhimit në vend se për komunitet teknike .

Tabela 1. Përformanca e sukseseve dhe mossukseseve të projekteve[1]

Activity	Successful Projects	Unsuccessful Projects
Sizing	Good	Poor
Planning	Very good	Fair
Estimating	Very good	Very poor
Tracking	Good	Poor
Measurement	Good	Very Poor
Quality control	Excellent	Poor
Change control	Excellent	Poor
Problem resolutions	Good	Poor
Risk analysis	Good	Very poor
Personnel management	Good	Poor
Supplier management	Good	Poor
Overall Performance	Very good	Poor

Tabela 1 rrjedh nga një nga libra, Patterns of Software System Failure and Success, botuar nga International Press Thomson. Shënimet e përformances se menaxherëve të projekte softuerike të suksesshme në krahasim me përformancen e tyre lidhur me anulimin e tejkalimeve të rënda. Përsosmëria në menaxhimin e projektit mund të bëjë më shumë për të rritur mundësinë e suksesit se çdo faktor tjetër, të tilla si blerja e mjeteve më të mirë, ose ndryshimin i gjuhëve të programuese .

[1] **JONES, CAPERS.** *Software Engineering Best Practices.* USA : The McGraw-Hill Companies, 2010 .

(Kjo është e vërtetë për aplikacione të mëdha mbi 1000 funksione. Për aplikacione të vogla në vargun e 100 funksioneve , aftësitë e inxhinierëve të softuerit vazhdojnë të mbizotërojnë rezultatet.)

Në tërësi, për përmirësimi e përformances së menaxhimit të projekteve softuerike mund të bëjë më shumë në mundësinë e optimizimit të suksesit të softuerëve dhe për të minimizuar shanset e dështim se në çdo aktivitet tjetër të njohur. Megjithatë, përmirësimi i përformances së menaxhimit të projekteve softuerik është një nga strategjitë më të vështira të përmirësimit .Një shumicë e dështimet e projekteve softuerike mund t'i atribuohen dështimet e menaxhimit të projektit në vend se për dështimet e stafit teknik. Për shembull, oraret e nënvlerësuar dhe kërkesat e burimeve është e lidhur me më shumë se 70 për qind e të gjitha projekteve që janë anuluar për shkak të tejkalimeve . Një tjetër problem i përbashkët i menaxhimit të projektit është i injoruar apo nënvleftësuar puna që lidhen me kontrollin e cilësisë dhe largimin e defekteve. Megjithatë, një tjetër problem i menaxhimit është dështim për t'u marrë me kërkesat e ndryshimeve në një mënyrë efektive

Duke pasur parasysh koston e lartë dhe vështirësi të rëndësishme që lidhen me sistemin e ndërtimit te softuerit, ju mund të mendoni se menaxherët e projektit softuerik do të duhej te jenë të trajnuar dhe të pajisur mirë me një nivel te lart (state-of-the-art) të planifikimit dhe mjeteve të vlerësimit, me analiza të konsiderueshme të strukturave të kostos historike të softuerit dhe me metodologji shumë të plotë në analizën e riskut. Këto janë supozime të natyrshme , por ato nuk janë të verteta. Menaxherëve të projekteve kryesore përdorin vetëm një shumëllojshmëri të gjerë të mjeteve të menaxhimit të projektit, por ata gjithashtu përdorin më shumë nga karakteristikat e këtyre mjeteve. Në pjesën për shkak të mungesës e përgatitjes akademike për menaxherët e projektit softuerike

Projekt menaxherët softuerik kanë qasje në mjete moderne të vlerësimit të kostos softuerike dhe më pak se 10 për qind kanë qasje në ndonjë sasi të konsiderueshme të dhënave të vlefshme historike nga projekte të ngjashme . Trajnime relativisht të dobëta dhe pajisjet e menaxhimit të projektit janë me të vërtet shqetësuese.

Ka një numër te madh te modeleve qe përdoren për vlerësim te kostos te tilla si :
COCOMO , KnowledgePlan , Price-S, SEER, SLIM dhe shumë te tjera te ngjashme .

Konceptin fillestar gjerë të menaxhim i projektit përfshinte të gjitha aktivitetet e nevojshme për të kontrolluar rezultatin e një projekti si :

- Madhësinë e projektit
- Vlerësimin kostove
- Planifikimin e orareve dhe objektivave
- Analizën e rrezikut
- Përcjelljen
- Zgjedhjen e teknologjisë
- Vlerësimin e alternativave
- Matjen e rezultateve

Ka 14 tema themelore që menaxherët e projektit duhet të dini , dhe për çdo temë është një temë me rëndësi për menaxhim profesional te projekteve softuerike :

1. Përmasat
2. Vlerësimin e projektit
3. Planifikimin e projektit
4. Metodologjia e përzgjedhjes
5. Përzgjedhjen e teknologjisë dhe e komponentëve
6. Kontrollin e cilësisë
7. Kontrollin e sigurisë
8. Supplier management
9. Progresi dhe përcjellja e problemeve
10. Standardizimi
11. Analizën e rrezikut
12. Analiza e vlerës
13. Vlerësimet e proceseve
14. Përmirësimet e proceseve

4. PROGRAMIMI DHE ZHVILLIMI I KODIT

Ky kapitull ka një tatëpjetë të pazakontë në mes temave të tjera, ajo merret me pyetje të rëndësishme që nuk janë të mbuluara mirë në literaturë inxhinierisë softuerike :

Pse ne kemi më shumë se 2500 gjuhë programimi?

Eksplorimi i popullaritetit të gjuhëve të programimit ?

Sa gjuhë të programimit vërtetë janë të nevojshme ?

Pse shumica e kërkesave moderne përdorin në mes 2 dhe 15 gjuhë të programuese ?

Cilat janë llojet kryesore të bugs dhe defekteve ndodhin në kodin burimor?

Këto tema nuk janë tema të vetëm që janë të rëndësishme në lidhje me programimin, por ata nuk janë diskutuar shpesh në revista inxhinieri softuerike apo libra.

4.1 HISTORI E SHKURTET MBI GJUHËT PROGRAMUESE

Është interesante të marrin në konsideratë historinë e programimit dhe zhvillimit të gjuhëve programuese . Esenca e programeve kompjuterik është për të kontrolluar sjelljen e një pajisje mekanike me anë të udhëzimeve diskrete që mund të jenë të ndryshme në mënyrë që të ndryshojë sjelljen e makinës .

Para Luftës së Dytë Botërore, një numër i kompanive në vende të ndryshme të ndërtuan pajisje elektro-mekanike kompjuterike, kryesisht për qëllime të veçanta të tilla si për llogaritur trajektoret ose zgjidhjen e detyrave matematikore . Por gjatë Luftës së Dytë Botërore, pajisjet kompjuterike janë zhvilluar me memorie ku mund të ruajë të dhëna dhe udhëzime .

Gjuhet më të hershme ishin të ruajtur në kompjuter si kodet binare ose gjuhë të makinës, të cilat padyshim ishin shumë të vështira për të kuptuar kodin ose të modifikoheshin . Vështirësia për të punuar me kodet e makinës çoi direkt në gjuhët që ishin më të përshtatshme për të kuptuar njeriu dhe të përkthehen në instruksionet e makinës.

Gjuhet me te hershme u quajtur gjuhë asambler dhe zakonisht kishin një korrespondencë një-me-një në mes udhëzimet të lexueshëm nga njerëzit (të quajtura kodi burimor) dhe udhëzimet e ekzekutueshme (i quajtur kodi objektit) . Por shpejtësia e kompjuterëve digjital shpejt rritet në vargje më të gjerë e aplikacioneve. Kur kompjuterët filluan të jenë qëllim në problemet e biznesit për të manipuluar tekstin dhe të dhëna, u bë e qartë se nëse kodi burimor përfshirë disa gjuhës dhe fjalor ne problem të domainit, pastaj gjuhë programuese do të jenë më të lehta për të mësuar dhe përdorur . Përdorimi i

kompjuterëve për të kontrolluar pajisjet fizike hapi një nevojë për optimizim të gjuhëve për t'u marrë me objekte fizike. Si rezultat, gjuhët programuese në domain të caktuar janë zhvilluar për qëllime dhe detyra të tilla si : biznesit, astronomisë dhe një mori të tjera .

4.2 PSE KEMI ME SHUMË SE 2500 GJUHË PROGRAMUESE ?

Koncepti i optimizimit të kodit burimor për lloje të veçanta të bizneseve apo problemeve teknike është një nga faktorët që çoi në përhapjen e madhe të gjuhëve programuese . Ka disa avantazhe teknike për të pasur gjuhët e programuese në përputhje me me fjalorin ne fusha të ndryshme të problemit. Gjuhët e tilla janë të lehtë për të mësuar për programuesit të cilët kanë njohuri në fusha të ndryshme. Zhvillimi i një gjuhe të re programuese që tërhoqi programuesit e tjera ka pasur edhe vlera shoqërore dhe prestigj. Si rezultat i arsyeve teknike dhe sociale, industritë e softuerit zhvilluara në gjuhë të reja programuese me frekuencë të habitshme. Sot, askush nuk e di se sa është numri aktual i gjuhëve programuese ,por listat e më të madhe në botuar e gjuhëve të programimit tani përmban 2500 gjuhë [2] .

Gjuhët e reja të programimi vazhdojnë të vijnë nga jashtë në një normë prej dy ose tre në muaj kalendarik, disa muaj më shumë se 10 gjuhë ndodhë të kenë ardhur. Një arsye për kaq shumë gjuhë për muaj është se një gjuhë e re mund të jetë zhvilluar nga një inxhinier softueri i vetëm për vetëm një apo dy muaj . U bë e qartë se mbajtja e evidencës së gjuhëve nuk do të jetë e shpejtë dhe e lehtë, por do të kërkojë përpjekje të vazhdueshme .

Është me interes sociologjik që pjesa më e madhe e gjuhëve janë zhvilluar . nga një individë apo ndoshta dy. Për shembull, Basic u zhvilluar nga John Thomas dhe Kurtz Kemeny, C është zhvilluar nga Dennis Ritchie , FORTRAN u zhvilluar nga John Backus , Java është zhvilluar nga James Gosling , dhe Objective C u zhvillua nga Brad Cox dhe Tom Love. Gjuhët e përgjithshme ishin zhvilluar zakonisht nga komitetet .

Për shembull, COBOL është zhvilluar nga një komision i njohur me të dhëna të madha nga Grace Hopper marinës amerikan .

[2] Lista Gjuheve nga Bill Kinnersley, http://people.ku.edu

Tabela 2 Kronologjia e zhvillimit të gjuhëve programuese [3]

1951 Assembly languages
1954 FORTRAN (Formula Translator)
1958 Lisp (List Processing)
1959 COBOL (Common Business-Oriented Language)
1959 JOVIAL (Jules Own Version of the International Algorithmic Language)
1959 RPG (formerly Report Program Generator)
1960 ALGOL (Algorithmic Language)
1962 APL (A Programming Language)
1962 SIMULA
1964 Basic (Beginner's all-purpose symbolic instruction code)
1964 PL/I
1964 CORAL
1967 MUMPS
1970 PASCAL
1970 Prolog
1970 Forth
1972 C
1978 SQL (Structured query language)
1980 CHILL
1980 dBASE II
1982 SMALLTALK
1983 Ada83
1985 Quick Basic
1985 Objective C
1986 C++
1986 Eiffel
1986 JavaScript
1987 Visual Basic
1987 PERL
1989 HTML (Hypertext Markup Language)
1993 AppleScript
1995 Java
1995 Ruby
1999 XML (Extensible Markup Language)
2000 C#
2000 ASP (Active Server Pages)
2002.NET

[3] **JONES, CAPERS.** *Software Engineering Best Practices.* USA : The McGraw-Hill Companies .

Sot në 2009, më shumë se 50 për qind e gjuhëve të programimit aktive janë object-oriented, ndërsa gjuhët e tjera janë gjuhët procedurale, gjuhë funksionale, ose që përdorin disa metoda të tjera të operacionit .

4.3 EKSPLORIMI I POPULLARITETIT TË GJUHËVE PROGRAMUESE

Ka një numër mënyrash të studimit të përdorimit dhe popullaritetin e gjuhëve të programimit. Këto përfshijnë :

1. Analiza statistikore e kërkimeve në internetit për gjuhë të veçanta .
2. Analiza statistikore e librave dhe artikujve të botuar në lidhje me gjuhët e veçanta .
3. Analiza statistikore e citimeve në literaturë në lidhje me gjuhë të veçanta .
4. Analiza statistikore e reklamimit të punës për programuesit që përmendin aftësi të gjuhës .
5. Analiza statistikore të gjuhëve në të gjitha kërkesat e trashëguara .
6. Analiza statistikore të gjuhëve të përdorura për aplikimet e reja .

Një kompani e quajtur Tiobe boton një analizë mujore të popullaritetit të gjuhëve programuese që rendit 100 gjuhë të ndryshme programimi. Nga kjo liste që është përpiluar në mars 2011 , 20 gjuhët më të popullarizuara janë të shënuara në tabelën 3

Tabela 3. Popullariteti i gjuhëve programuese [4]

1. Java
2. C
3. C++
4. C#
5. PHP
6. Python
7. (Visual) Basic
8. Objective-C
9. Perl
10. JavaScript
11. Ruby
12. Assembly*
13. Delphi
14. Go
15. Lisp
16. Lua
17. Ada
18. Pascal
19. NXT-G
20. Scheme*

[4] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Popullariteti i gjuhëve të programimit ka një ngjashmëri të caktuara me popullaritetit e televizioneve . Disa shfaqje të reja tërheqin miliona shikues, dhe mund të zgjasin disa sezona . Po ashtu edhe disa gjuhë programimi kanë jetë aktive vetëm disa sezona dhe pastaj zhduken. Disa gjuhë të tjera bëhen standarde dhe mund të zgjasin për shumë vite. Megjithatë,kur të gjitha gjuhët janë konsideruar 2500, jeta mesatare e një gjuhë programuese aktive që është duke u përdorur për zhvillim të ri është më pak se pesë vjet .

4.4 SA SHUMË GJUHË PROGRAMUESE VERTET JANË TË NEVOJSHME ?

Numri i madh i gjuhëve të programimit ngre pyetje themelore që duhet t'i adresohen literaturës së inxhinierisë softuerike :

Duke pasur mijëra gjuhëve programuese ngre një pyetje logjike : A është ekzistenca e më shumë se 2500 gjuhëve programuese një gjë e mirë apo një gjë e keqe ? Argumenti që pohon që ka mijëra gjuhë është fakti i mirë se gjuhët kanë tendencë të jenë të optimizuara për klasat unike të problemeve. Për probleme e reja që kanë hasur, ata kërkojnë gjuhëve të reja të programimit, ose të paktën kjo është një hipotezë.

Argumenti që pohon që ka mijëra gjuhë është fakt i keqe për arsye ekonomisë .Nevojë të vazhdueshme për të trajnuar programuesit në zhvillimet e gjuhëve të fundit është i shtrenjtë. Shumë mjete të dobishme si mjete të analizave statike dhe mjetet e testimeve të automatizuar mbështesin vetëm një numër të vogël të gjuhëve të programimit, dhe për këtë arsye mund të kërkojnë modifikime të shtrenjta për gjuhë të reja .

Ekzistenca e mijëra gjuhëve të programimit ka krijuar një sub industri brenda inxhinierisë softuerike . Kjo subindustri e re ka të bëjë me përkthimin gjuhë të vdekura ose atyre qe po vdesin në gjuhë të reja . Logjikisht subindustria është për rinovim ose kryerjen periodike të aktivitete të veçanta të mirëmbajtjes në aplikacionet e trashëguara për të pastruar nga kodi i vdekur, heqjen e moduleve të gabueshme, dhe për të zvogëluar rritjen e pashmangshme në kompleksitet e domosdoshme që ndodhin me kalimin e kohës për shkak të ndryshimeve të vogla që përsëriten. Linguistët dhe ata që njohin mirë gjuhët natyrore të njeriut janë të vetëdijshëm se përkthimi nga një gjuhë në një tjetër nuk është i përsosur. Për t'u marrë me çështjen se sa shumë gjuhë të programimit janë të nevojshme, është e dobishme për të filluar duke e konsideruar gjithësinë e zonave problem që duhet të vihet mbi kompjuterët.

Këto dy kategori të përgjithshme pasqyron format e madhe e softuerike të cilat në të vërtetë ekzistojnë sot:

1. Softuerët për proceset e informacionit, dhe

2. Softuerët qe kontrollojnë pajisjet fizike ose merren me vetitë fizike të tilla si zërit ose dritën .

Këto dy kategori të gjera mund të çojë në përfundimin se ndoshta dy gjuhëve të programimit do të jetë një numër minimal që do të jenë në gjendje për t'i trajtuar të gjitha problemet. Një gjuhë do të jetë e optimizuar për sistemet e informacionit, dhe një tjetër do të jetë e optimizuar për pajisjet fizike dhe sinjalet elektronike.

4.4.1 DOMENI I PROBLEMEVE TË APLIKACIONET SOFTUERIKE

Domeni i problemeve Logjike dhe matematike

1. Llogaritjet matematikore
2. Logjika dhe shprehjet algorithmetike
3. Të dhënat numerike
4. Tekstet dhe të dhënave
5. Koha dhe datat

Domeni i problemeve Fizike

1. Sinjalet bazë me sensor elektronik .
2. Sinjalet akustike dhe me zë
3. Imazhet Statike
4. Imazhet e lëvizshme apo dinamike
5. Ngjyrat

Nëse permutacionet e dhjete probleme të lartcekura do t'i merrnim parasysh atëherë ne mund të përfundojmë në fund me 3.628.800 gjuhë programuese . Kjo ka më shumë gjasa të ndodhë se sa një "super gjuhe" që mund t'i trajtoj të gjitha fushat . Ne qoftë se këto dhjetë fusha do t'i minimizonim në katër fusha do të përfundonim me rreth 5040 gjuhë programuese të ndryshme. Në norma e gjuhëve të reja po ndodh që të rritet prej rreth 100 në vit, Nga pikëpamja ekonomike, kjo nuk duket të jetë një zgjidhje me kosto efektive .

Duke supozuar se inxhinieret e softuerit do të arrijë në 5.040 gjuhë, shpërndarja e mundshme e këtyre gjuhëve do të ishte :

- ✓ 4800 gjuhë do të jenë të vdekur ose duke vdekur , me disa programmer .
- ✓ 200 gjuhë do t'i plotësojnë të gjitha kërkesat dhe për këtë arsye do të duhej t'i mirëmbajmë .
- ✓ 40 gjuhë do të jenë të reja dhe grumbullimi i numrit në rritje i programuesve .

4.5 PSE SHUMICA E APLIKACIONEVE PËRDORIN PREJ 2 DERI NE 15 GJUHË PROGRAMUESE?

Një fenomen i mrekullueshëm i inxhinierisë softuerike është prania e gjuhëve programuese të shumta në të njëjtin aplikacion. Kjo nuk është një prirje e re, dhe aplikacioneve shumë të vjetra përdornin kombinime të tilla si COBOL dhe SQL. Më shumë kombinime të kohëve të fundit mund të përfshijë Java dhe HTML apo XML. Një fenomen i ngjashëm është fakti se shumë gjuhë të programimit janë vetë kombinimet e dy ose më shumë gjuhëve të programimi tjera.

Për shembull, gjuha Objektivi C kombinon tiparet nga Smalltalk dhe C. Gjuha Ruby kombinon tiparet nga Ada, Eifel, Perl, Python dhe të tjera. Të kujtojnë se shumica e gjuhëve të programimit janë të specializuara deri diku, dhe këto duken të jenë më të popullarizuara se gjuhët me qëllime të përgjithshme . Një hipotezë që shpjegon pse aplikacionet përdorin disa gjuhë të ndryshme programimi është se "hapësira e problemit" në zbatim është më i gjerë se "hapësira e zgjidhjeve e gjuhëve të programuese individuale.

Megjithatë, shumë gjuhë programuese duket të jenë i optimizuar vetëm nga 1-3 fusha të problemit. Kjo krijon një situatë ku shumë gjuhë programuese janë të nevojshme për të zbatuar të gjitha fushat e problemit në aplikim .Pasojat e shumë gjuhëve të ndryshme në të njëjtën aplikacione bënë që zhvillimi të jetë më i vështirë, debugging është më i vështirë, analiza statike është më e vështirë, dhe inspektimit kodi është më i vështirë, po ashtu mirëmbajtjen dhe zgjerimi janë më të vështira .

4.6 ÇFARË LLOJE TË BUGS APO DEFEKTEVE NDODHIN NË KODIN BURIMOR

Në vitin 2008 dhe 2009, një studim i ri dhe i madh ishte kryer për të identifikuar 25 bags apo defektet me të rënda dhe me të zakonshme të softuerëve . Ky studim u financua nga Instituti SANS , me bashkëpunimin të MITRE dhe rreth 30 organizatave të tjera. Ky studim i ri tërheq një vëmendje të madhe . Në historinë e cilësisë së programeve dhe të sigurisë, ajo pa dyshim do të renditet si një raport historik . Në të vërtetë, të gjitha grupet inxhinierisë softuerike duhet të marrë kopjet e raportit dhe të bëjnë të preferuar leximin për inxhinierët e softuerit, personelin e sigurimit të cilësisë, dhe gjithashtu për menaxherët dhe drejtuesit softuerik .

Në raport mund te qaseni përmes Institutit SANS[5] apo ueb faqet Mitre [6].

Me gjithë faktin se inxhinieria softuerike tani është një profesion i madh dhe me miliona aplikacione janë koduar, vetëm kohët e fundit ka pasur një përpjekje serioze dhe të përqendruar për të kuptuar natyrën e të metave dhe defekte që ekzistojnë në kodin burimor.

[5] http://www.sans.org/
[6] http://www.cwe-mitre.org

Raporti i sans është e rëndësishme sepse lista e 25 problemeve serioze është zhvilluar nga një grup prej rreth 40 ekspertë nga organizatat e softuerike të mëdha. Si rezultat, është e qartë se problemet e përmendura janë problem universale të programimit dhe jo çështje për një kompani të vetme .

Raporti i sans është një shembull shumë inkurajuese të llojit të progresit që mund të bëhet kur ekspertët e lartë nga shumë kompani të punojnë së bashku në një mënyrë bashkëpunuese për të shqyrtuar problemet e përbashkëta. Grupi i sans studimi përfshinte ekspertë nga akademia, qeveria, dhe shoqëritë tregtare. Ajo është gjithashtu inkurajuese që këto tri lloje të organizatave të ishin në gjendje të bashkëpunojnë me sukses. Marrëdhëniet normale midis tyre ishin shpesh kundërshtuese dhe jo bashkëpunuese, në mënyrë që të gjitha të punojnë së bashku dhe të prodhojnë një raport të dobishëm është një dukuri mjaft të e rrallë.

Disa prej organizatave që morën pjesë në studim sans të përfshijë në rend alfabetik :

- Apple
- Aspect Security
- Breach Security
- CERT
- Homeland Security
- Microsoft
- Mitre
- National Security Agency
- Oracle
- Perdue University
- Red Hat
- Tata
- University of California

Lista e përgjithshme e 25 problemeve të sigurisë ishte e ndarë në tre zona të mëdha aktuale. Lexuesit i kërkohet për të shqyrtuar raportin e plotë .

5. ORGANIZIMI DHE SPECIALIZIMI I SOFTUERIT NE GRUP.

Më shumë se pothuajse në çdo fushë tjetër teknike apo inxhinierike , zhvillimi i softuerëve varet nga mendja e njeriut mbi përpjekjet e njeriut dhe me organizimet e njerëzve .Që nga dita ku një projekti fillon derisa ai të pensionohet , ndoshta 30 vjet më vonë përfshirja e njeriut është kritike për çdo hap të zhvillimit, zgjerimit, mirëmbajtjes dhe mbështetjes së klientëve. Kërkesat e softuerit janë nxjerrë nga diskutimet e njeriut për aplikacionet e se ardhmes. Arkitektura e softuerit varet nga njohurit e njerëzve specialistë. Dizajnimi i softuerëve është e bazuar në të kuptuarit e njeriut shtuar nga mjetet që merren ne disa aspekte mekanike, por asnjë nga aspektet intelektuale.Kodi i softuerit është shkruar rresht për rresht nga njerëzit zanatli dhe përfshin sasi më të lartë të përpjekjeve të njeriut të çdo produkti modern te prodhuar. Edhe pse mjetet e automatizuara të analizave statike dhe disa formave të testim të automatizuar ekzistojnë, mendja e njeriut është gjithashtu një mjet primar për gjetjen bagave dhe defekte të sigurisë. Për fat të keq, aspekti i cilësisë dhe i sigurisë mbeten lidhje të dobëta të softuerit .

Ajo mund të jetë se recesioni global që do të japë një nxitje të fortë për të filluar për të migruar nga zhvillimi me porosi në ndërtimi të komponentëve standarde të ripërdorueshme Recesioni global mund të sigurojë motivimin për hartimin e programeve më të sigurta dhe me cilësi të lartë, dhe për të lëvizur drejt niveleve më të larta të automatizimit në kontrollin e cilësisë dhe të sigurisë. Pavarësisht nga faktit se softueri ka përmbajtje më të lartë të punës se çdo produkt i prodhuar, struktura e organizimit për zhvillimin e softuerit ne grup nuk mbulohet mirë në literaturën softuerike. Ka raporte anekdotike për vlerën e temave të tilla si pair-programming, small self-organizing teams, Agile teams, colocated teams, matrix versus hierarchical organizations, project offices, dhe disa te tjera . Por këto raporteve kane mungesës të sasisë së rezultateve. Një nga koleksionet më të madhe të informacionit lidhur me ndërlidhjen e ekipit që është në dispozicion për publikun e gjerë është e vendosur të raporteve dhe të dhënave të botuara nga International Software Benchmarking Standards Group (ISBSG) .Shumë kompani të mesme dhe të mëdha të tilla si bankat dhe kompanitë e sigurimit kanë vetëm IT organizatat . Ndërsa ka probleme organizative brenda kompani të tilla si : Apple, Cisco, Google, IBM, Intel, Lockheed, Microsoft, Motorola, Oracle, Raytheon, SAP dhe të ngjashme, të cilat zhvillojn sisteme dhe embedded softuere , si dhe IT softuare . Brenda shumicës se kompanive që ndërtojnë IT softuerë dhe sisteme softuerike të dy organizimet janë krejtësisht të ndryshme. Normalisht, IT e kanë te organizuar raportimin të zyrtari i lartë informacioni (chief information officer CIO) , ndërsa Grupet e sistemeve softuerike zakonisht raportojnë të zyrtari më i lartë i teknologjisë (chief technology officer CTO). CIO dhe CTO janë zakonisht në të njëjtin nivel, kështu që asnjeri nuk ka autoritet mbi të tjetrin .

6. QASJET DHE METODOLIGJIT E ZHVILLIMIT TË SOFTUERIT

Njëra prej qasjeve për zhvillimin e softuerëve kompleks është DDD (Domain Driven Design).

DDD nuk është një teknologji apo metodologji. DDD siguron një strukturë të praktikave dhe të terminologjisë për marrjen e vendimeve të dizajnimit që të përqendrohet në përshp-ejtimin e projekteve softuerike që kanë domain kompleks . DDD-ja është thellësisht e lidhur me implementimin e një modeli në zhvillim të koncepteve kryesore të biznesi .

DDD përfshin :

- ✓ Fokusi kryesor në projektet në thelbin dhe logjiken e domenit
- ✓ Mbështet dizajnë kompleks të modelit
- ✓ Inicimin e krijimit te bashkëpunimit ne mes eksperteve teknik dhe eksperteve të domenit (fushës qe projekti do te zhvillohet) për ta konceptualizuar problemin gjithnjë më në thelb.

Përkufizimet bazë

Domain - Një sferë e diturisë, ndikimit, apo aktiviteve . Fushë për të cilën përdoruesit aplikojnë një program .

Modeli - Një sistem i abstraksioneve që përshkruan aspektet e një domeni dhe mund të përdoret për të zgjidhur problemet që lidhen me atë domain .

Zhvillimit i softuerëve shumë shpesh aplikohet për automatizimin e proceseve që ekzistojnë në botën reale, ose për të ofruar zgjidhje për problemet reale të biznesit . Kur proceset e biznesit apo problemeve të botes reale fillojnë të automatizohen softuerët përfshin një domein (fushë) të programeve .

Ne duhet të kuptojmë që nga fillimi se softuerët janë te orientuar dhe kanë lidhje të forte me domenin. Softueri është i përbërë nga kodi. Ne mund të shpenzojmë shumë shumë kohë me kodin, dhe e shohim thjesht softuerin si objekte dhe metoda.

Konsideroni prodhimin e një makine . Punëtorët e përfshirë në prodhim mund të specializohen në prodhimin e pjesë të makinave, por duke bërë kështu që ata shpesh kanë një pamje të kufizuar të të gjithë procesin e prodhimit të makinave . Ata fillojnë të shikojnë makinën si një koleksion të madh të pjesëve të cilat duhet të përshtaten së

bashku, por një makinë është më shumë se kaq . Një makinë e mirë fillon me një vizion, ajo fillon me specifikimet e shkruar me kujdes dhe vazhdon me dizajnim shumë dhe shumë e projektimit muaj ndoshta vite të kohës shpenzohen në hartimin, ndryshimin dhe rafinimit deri sa të arrijnë përsosmëri deri sa pasqyron vizionin origjinal. Përpunimi i projektit nuk është i gjithi në letër. Pjesa më e madhe përfshin dizajnimi e modele të makinës, dhe testimi i tyre në kushte të caktuara për të parë a punojnë . Dizajni është ndryshuar shumë herë në rezultatet e testimit makinë është dërguar në prodhim në fund dhe pjesët janë krijuar dhe mbledhur së bashku.

Zhvillimit i softuerëve është i ngjashëm . Ne nuk mundem vetëm të ulemi edhe të kodojmë nuk duhet ta bëjmë këtë edhe pse kjo mund të punon në raste të pa rëndësishme por nuk mund të krijojmë softuer komplex. Në mënyrë që të krijojë softuerë të mirë, ju duhet të dini se çfarë softueri është mbi të gjitha. Ju nuk mund të krijoni një sistem softuerik bankar nëse nuk keni kuptuar domenin e bankës .

Shembull :

A është e mundur për të krijuar softuerë komplekse bankar pa njohuri të mirë për domenin ?

- *Jo, në asnjë mënyre.*

Kush i njeh bankat ? Arkitektet e softuerit ?

- *Jo, Jo . Ata vetëm përdorin bankën për të mbajtur paratë të sigurta dhe në dispozicion, kur ata ka nevojë për to.*

Analistet e Softuerit ?

- *Jo me të vërtetë. Ata din për të analizuar një temë të caktuar, kur i është dhënë i gjithë materiali i nevojshëm .*

Developerat (Zhvilluesit e programit) ?

- *Jo Jo harroni ata , kush tjetër .*

Bankierët ?

- *Po pikërisht . Sistemin bankar shumë mirë e njohin personat që punojnë aty qe janë specialist. Ata e dine të gjitha detajet, të gjitha rastet, të gjitha çështjet e mundshme, të gjitha rregullat.*

Kjo është ajo ku ne gjithmonë duhet të fillojmë: domenin .
Si mund ta bëjë të aftë softuerin ne harmoni me domenin?
Mënyra më e mirë për ta bërë këtë është të bëjë një reflektim softuerik të domenit. Softuerik duhet të përfshijnë konceptet kryesore dhe elementet e fushës, dhe për të realizuar pikërisht marrëdhëniet në mes tyre. Softueri ka për model domenin.

Pra, ne fillim me domenin. Pastaj çfarë? Një domein është diçka i kësaj bote . Ajo nuk mund vetëm të merren dhe ta derdhim mbi tastierë e kompjuterit e të bëhet kod. Ne kemi nevojë për të krijuar një abstraksion të domeint. Ne mësojmë shumë ,ne lidhje me domeinin duke folur me ekspertët e fushës. Por kjo njohuri nuk do të transformohet lehtë në ndërtim të softuerit nëse ne nuk ndërtojmë një abstraksion të tij, një plan

në mendjet tona. Në fillim, projekti është gjithmonë jo i plotë. Por më kohë, duke punuar në të ne kemi bërë atë të mirë dhe ajo bëhet gjithnjë e më e qartë për ne.

Çfarë është ky abstraksion? Ai është një model i domeinit .

Sipas Eric Evans[7], një model i domenit nuk është një diagram i veçantë, ajo është ideja se si duhet të përcillet .

Modeli është një pjesë thelbësore e dizajnimit te softuerit . Të gjitha të menduarit tone rreth procesit në lidhje me domenin paraqitet në këtë model. Ne kemi nevojë për të komunikuar këtë model me ekspertë të fushës, me dizajnerët , dhe me zhvilluesit . Model është thelbi i softuerit , por ne kemi nevojë për të krijuar mënyra të për të shprehur atë për të komunikuar me të tjerët.

Ne nuk jemi të vetëm në këtë proces, kështu që ne duhet që të ndajnë njohuritë dhe informacionet dhe ne duhet të bëjmë atë mire në mënyrë precize dhe qartë. Kemi mënyra të ndryshme për të bërë këtë p.sh : me diagrame grafike, use case, vizatime , fotografi, etj. Kur ne kemi një model të shprehur, ne mund të fillojnë të bëjnë dizajnimin e kodit . Kjo është e ndryshme nga dizajnimi i softuerit . Dizajnimi i softuerit është si krijimi i arkitekturës së një shtëpie. Nga ana tjetër, dizajnimi i kodit është punë mbi detajet si vendndodhjen e një pikturë në një mur të caktuara. Dizajnimi i kodit është gjithashtu shumë i rëndësishme, por jo themelor si dizajnimi i softuerit . Një gabim ne dizajnimin e kodit është zakonisht më i lehtë të korrigjohet, ndërsa gabime ne dizajnimin e softuerit janë shumë më të kushtueshme për tu riparuar. Megjithatë produkti përfundimtar nuk do të jetë e mirë pa dizajn të mirë të kodit.

Ka metodologji të ndryshme për dizajnimin e softuerëve .

Disa nga ato janë :

- ✓ Metodologjia Ujëvara
- ✓ Metodologjia Agile .

DDD kombinon dizajn me zhvillim të praktikës, dhe tregon se si dizajn dhe zhvillimi mund të punojnë së bashku për të krijuar një zgjidhje të mirë. Dizajni i mirë do të përshpejtojë zhvillimin, ndërsa reagimet qe vijnë nga procesi i zhvillimit do të rrisin dizajnin .

[7] http://domaindrivendesign.org/about

6.1 METODOLOGJIA UJËVARA

Historiku

Modeli Ujëvara nuk është një model i ri , ky model ka filluar të përdoret në vitet e 1970 nga Royce[8] . Emri Ujëvara e ka marr si rezultat i rrjedhjes neper procese dhe kjo rrjedhje nuk është e kthyeshme .

Kjo metodë përfshin 5 faza :

1. Analizën e Kërkesave
2. Dizajnimin
3. Implementimin
4. Testimin
5. Mirëmbajtjen

Ekspertë të biznesit vënë një sërë kërkesash të cilat komunikojnë me analistët e biznesit. Analistët të krijojë një model të bazuar në këto kërkesa, dhe i kalojnë rezultate e tyre të zhvilluesit, të cilët fillojnë kodimin bazuar në atë që ato kërkesa kanë marrë . Kjo është njëra prej rrjedhjeve . Ndërsa kjo ka qenë një qasje tradicionale në dizajnimin e softuerit dhe është përdorur me një nivel të caktuar të suksesit gjatë viteve ajo ka defekte e saj dhe kufijtë. Modeli ujëvara është paraqitur ne figurën 1.

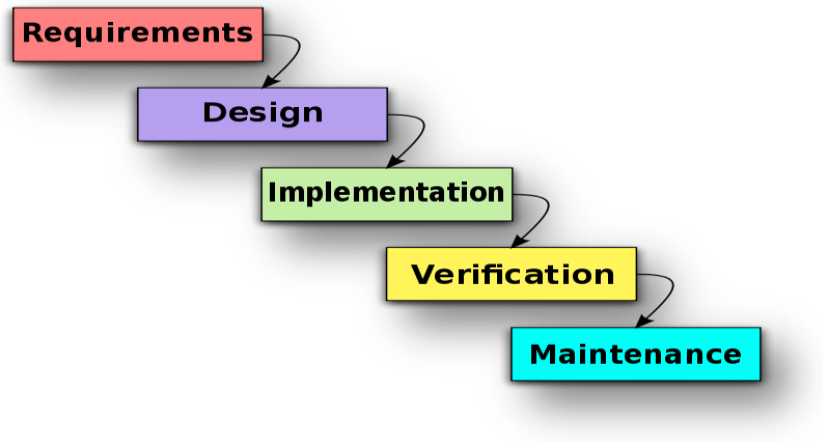

Figura 2 Metodologjia Ujëvara[9]

Përparësitë

Edhe pse modeli ka kritikë, ai mbetet ende i dobishëm për lloje të caktuara të projekteve dhe nëse zbatohen siç duhet mund të prodhojë kursime të konsiderueshme të kostos dhe të kohës Nëse ju duhet të përdorni apo jo varet shume se si i kuptoni kërkesat e klientit tuaj, dhe sa cilat janë rrezikshmëri që ju presin qe projekti të përparon.

[8] http://en.wikipedia.org/wiki/Winston_W._Royce
[9] http://en.wikipedia.org/wiki/File:Waterfall_model.svg

Mangësitë
Problemi kryesor është se nuk ka feedback nga analistët me ekspertë të biznesit ose nga zhvilluesit të analistët .

6.2 METODOLOGJIA AGILE

Metodologjia agile është një grup i metodologjive të zhvillimit të softuerit në bazë të zhvillimit përsëritës dhe në rritje, ku kërkesat dhe zgjidhjet zhvillohen përmes bashkë-punimit në mes të vetë-organizimit të ekipeve të funksioneve ndërlidhëse .

Karakteristikat

Ekzistojnë shumë metoda të veçanta të zhvillimit agile . Me te promovuarat janë :

- ✓ Puna ekipore,
- ✓ Bashkëpunimi dhe
- ✓ Procesi i përshtatshmërisë në të gjithë ciklin te jetës së projektit.

Metodat agile ndajnë detyrat në pjesë vogla me planifikime minimale, dhe nuk përfshijnë planifikim afat gjatë. Përsëritjet janë korniza të shkurtra kohore që zakonisht zgjasin 1-4 javë.

Çdo ripërsëritje që bënë ekipi punues nëpër një cikël të plotë të zhvillimi softuerik përfshirë :

- ✓ Planifikimin
- ✓ Analizën e kërkesave
- ✓ Dizajnimin
- ✓ Kodimin
- ✓ Testimin e njësive,
- ✓ Testimin e pranimit, kur produkti i demonstrohet palëve të interesuara .

Kjo minimizon rrezikun e përgjithshëm dhe lejon që projekti t'iu përshtatet ndryshimeve shpejt .

Në figurën është paraqitur metodologjia agile .

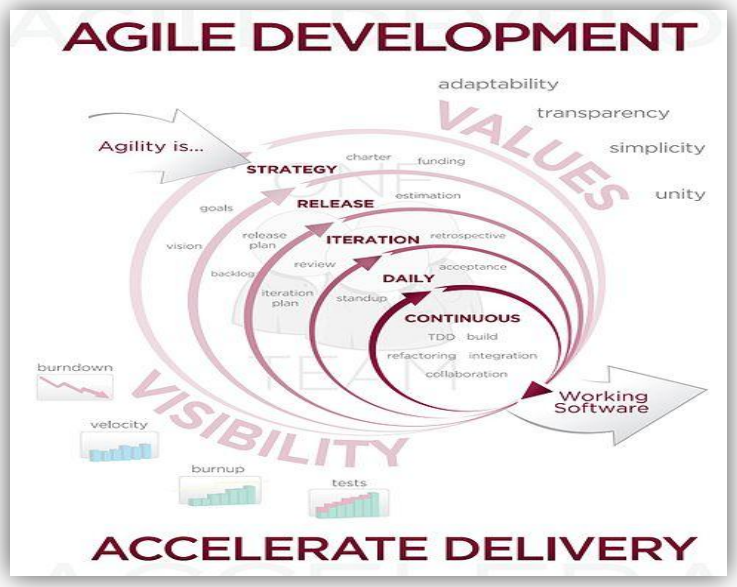

Figura 3 Metodologjia Agile [10]

Metodat më të njohura te zhvillimit të softuerit ne agile janë :

- ✓ Agile Modeling
- ✓ Agile Unified Process (AUP)
- ✓ Dynamic Systems Development Method (DSDM)
- ✓ XP (Extreme Programming)
- ✓ Essential Unified Process (EssUP)
- ✓ Feature Driven Development (FDD)
- ✓ Open Unified Process (OpenUP)
- ✓ Scrum
- ✓ Velocity tracking

Përparuesit

- ✓ Klienti ndahet i kënaqur me shpërndarjen e shpejtë të programit
- ✓ Kërkesat për ndryshime janë te mirëseardhura edhe ne fazën e fundit te zhvillimit.
- ✓ Shpejtësia e dorëzimit te softuerit (pas një apo disa javë në vend se muajve)

[10] http://www.gantprojects.com/wp-content/uploads/2011/05/agile1.jpg

- ✓ Zhvillimi i qëndrueshëm i aftë për të mbajtur një ritëm konstant
- ✓ Bashkëpunim të ngushtë të përditshme mes njerëzve të biznesit dhe zhvilluesve .
- ✓ Bisedë ballë për ballë është forma më e mirë e komunikimit
- ✓ Projektet janë ndërtuar rreth individëve të motivuar, të cilëve duhet t'i besohet
- ✓ Vëmendje të vazhdueshme për përsosmëri teknike dhe dizajn të mirë
- ✓ Thjeshtësi
- ✓ Vetë-organizimi i ekipeve
- ✓ Përshtatja e rregullta ne rast të ndryshimit te rrethanave

Mangësitë

- ✓ Intensiteti i punës mund të çojë ekipin në Burnout[11]
- ✓ Të menduarit strategjik mund të jetë e vështirë që të përshtaten në "orar"
- ✓ Proceset përsëritëse marrin një përqindje të madhe të kohës në përsëritje
- ✓ Duke pritur për resurset njerëzore të ekipeve të jashtme mund të rris më shumë gjasa për punë përsëritëse .
- ✓ Riprodhushmeria e vazhdueshme e bërë nga zhvilluesit e pa dizajnim të plote do të prodhojë kod që është e vështirë për t'u kuptuar apo ndryshuar .

6.2.1 XP (Extreme Programming)

XP është një nga proceset me të popullarizuara të metodologjisë agile .
Këto metodologji janë një lëvizje kolektive kundër modelit te qasjes ujëvarë që rezultojnë nga vështirësitë e duke u përpjekur për të dalë me të gjitha kërkesat ne fillim veçanërisht në pjesën e ndryshimit të kërkesave . Është me të vërtetë është vështirë për të krijuar një model të plotë i cili mbulon të gjitha aspektet e një domeini ne fillim .

Projekti i pare ne XP ka filluar me 6 mars 1996.

Extreme Programim fillon me pesë vlera :

- ✓ Komunikimi,
- ✓ Feedback,
- ✓ Thjeshtësi,
- ✓ Guximi, dhe
- ✓ Respekti

[11] http://en.wikipedia.org/wiki/Burnout_(psychology)

Ajo pastaj elaboron këto në 14 parime dhe 24 praktika. Ideja është se praktikat janë gjëra konkrete që një ekip mund të bëjë ditë-për-ditë , ndërsa vlerat janë të njohurive themelore dhe të kuptuarit që mbështesin qasjen . Vlerat pa praktika janë të vështirë për tu aplikuar dhe mund të aplikohet në shumë mënyra që është e vështirë të dihet ku të fillojë. Praktikat pa vlera janë aktivitetet rutinë, pa një qëllim. Të dyja vlerat dhe praktikat janë të nevojshme . Shumë nga praktikat e XP-së janë të vjetra tashmë janë provuar të jetë shumë të suksesshme në shumë kompani të madhësive të ndryshme dhe të industrive të gjera botërore.

Extrem programimi është i suksesshëm, sepse ai thekson kënaqësinë e konsumatorit. Ekstreme Programim fuqizon zhvilluesit t'iu përgjigjur kërkesave për ndryshim edhe në fund të ciklit të jetës. Ekstreme Programim thekson punën në grup. Menaxherët , konsumatorët, dhe zhvilluesit janë të gjitha partnerë të barabartë në një ekip bashkëpunuese. XP zbaton një ambient të thjeshtë, por efektive duke bërë të mundur ekipi të bëhet shumë produktiv . Ekipi vetë-organizohet rreth problemit për të zgjidhur atë ne mënyrë sa më efikase që të jetë e mundur.

XP - Programmeret vazhdimisht komunikojnë me klientët e tyre dhe bashkëpunëtoret e tjerë programmer . Ata marrin feedback nga testimi i programeve të tyre duke filluar nga dita e pare. Ata ofrojnë sistemin të konsumatorët sa më shpejt që të jetë e mundur dhe implementojn ndryshimet e sugjeruara . Çdo sukses i vogël thellon respektimin e tyre për kontributin e veçantë të secilit anëtare dhe te grupit. Ekipet e XP-se janë në gjendje dhe kane guxim t'iu përgjigjur kërkesave të ndryshme edhe ne teknologji .

7. ZHVILLIMI DHE MIRËMBAJTJA E SOFTUERVE NË VITIN 2049

Nga vitet 1960 deri 2009, zhvillimin e programeve ka qenë në thelb një zeje ku aplikacionet të ndërlikuara janë dizajnuara si objekte unike dhe ndërtuar pastaj nga kodi burimor të bazuar në rreshta . Kjo metodë e zhvillimit e cila zakonisht shkruhet rresht për rresht kurrë nuk mund të jetë efikase, ekonomik, apo të arritur nivele të qëndrueshëm të cilësisë dhe sigurisë . Shpresojmë, nga 2049, nga disiplina e vërtetë inxhinieri do të lejojë që të zhvillohen softuer nga një formë e shprehjes artistike në një disiplinë të fortë inxhinierike.

Ky seksion paraqet një analizë hipotetike te aplikacioneve softuerike mënyrën që do të mund të projektohen dhe ndërtohen softuerët në vitin 2049 . Nëse softuerët duhet të bëhen një disiplinë e vërtetë inxhinierike, atëherë shumë më tepër se zhvillimi i kodi duhet të përfshihen : arkitektura , kërkesat, dizajnimi , zhvillimin e kodit, mirëmbajtje, mbështetjen e konsumatorëve, trajnime, dokumentacioni, menaxhimi i projektit, siguria, cilësia, kontrolle të ndryshme të, standardeve, dhe shumë tema të tjera duhet të merren parasysh . Pika e fillimit në vitin 2009 dhe 2049 natyrisht do të jetë e kërkesa për aplikim te ri . Në 2009 përdoruesit janë intervistuar për zhvillimin e kërkesave për aplikacione të reja, por në 2049 një metodë të ndryshme mund të jenë të në dispozicion . Le të supozojmë se kërkesa që do të zhvillohen rreth 2049 është një formë të re të planifikimit softuerëve dhe me kosto të vlerësuar të mjeteve.

Ky mjet do të japë vlerësimet e kostos të softuerëve, vlerësimet e organizimit, vlerësime të cilësisë, dhe vlerësimet e personelit si të bëjë një numër të mjeteve ekzistuese.

Megjithatë, do të futën një numër i karakteristikave të reja, të tilla si:

1. Fillimisht një periudhë kohore për njohjen e kërkesave të plota
2. Vlerësimet e kërkesave të ndryshme gjatë zhvillimit
3. Vlerësimet e sasisë së defekteve te zvarritura në kërkesa
4. Integrimi i analizës se riskut
5. Integruarimi i analizës vlerësuese .
6. Integrimi i analizës se sigurisë .
7. Parashikim të efekteve të çdo niveli mbi produktivitetin dhe cilësinë
8. Parashikim i efekteve të sasive të ndryshme të materialeve të ripërdorueshme
9. Parashikim i pasojave të agjentëve inteligjente të zhvillimit të softuerit
10. Parashikim i pasojave të agjentëve inteligjente për mirëmbajtjen e softuerit
11. Parashikim i pasojave të agjentëve inteligjente në dokumentacionin softuerik
12. Parashikim i efekteve të agjentëve inteligjente në mbështetje të konsumatorëve
13. Parashikim i pasojave të agjentëve inteligjente për dështimet softuerike

14. Automatizimin e konvertimit ndërmjet pikave të funksionit, LOC, pika te treguara dhe kështu me radhë .
15. Vlerësimet e grafikoneve të mësuarit nga ana e përdoruesve të aplikacionit
16. Parashikimi i gabimeve të bëra, derisa përdoruesit të mësojnë aplikacionin
17. Vlerësimet e kujdesit ndaj klientit dhe të mirëmbajtjes për 10 vjet pas vendosjes
18. Parashikimi i rritjes së kërkesës për 10 vjet pas vendosjes fillestare
19. Integrimi për kapjen e të dhënave historike, gjatë zhvillimit dhe mirëmbajtjes
20. Krijimin automatik te standardeve të produktivitetit dhe cilësisë
21. Këshilla të ekspertëve për kontrollin e cilësisë së programeve
22. Këshilla të ekspertëve për të kontrollit të sigurisë së softuerëve
23. Këshilla të ekspertëve për qeverisjen e softuerit.
24. Këshilla të ekspertëve për pronësinë intelektuale
25. Këshilla të ekspertëve mbi standardet dhe rregulloret përkatëse

Karakteristika e shekullit 19 dhe 20 dhe të rejat të vlerësimit të mjeteve do të përfshijë krijimin e një licence të përgjithshëm te International Software Benchmarking Standards Group (ISBSG), në mënyrë që klientët do të jenë në gjendje të përdorin mjet për të mbledhur dhe analizuar provat e aplikimeve të ngjashme, përderisa vlerësojnë aplikimet e reja. Çdo klient do të duhet të paguajnë për këtë shërbim, por ajo mund të integrohet në mjetet e veta . Kështu, jo vetëm që do të vlerësohen të prodhohen nga mjet, por edhe standarde për aplikime të ngjashme do të grumbullohen dhe përdoren për të mbështetur vlerësim, duke siguruar të dhëna historike në lidhje me mjet të ngjashëm te aplikacionit . Rivlerësimi ka për qëllim të përdoret për të mbledhur të dhëna historike dhe të krijojnë standardeve gjysmë automatike

Pyetjet do të përfshin tema të veçanta të tilla si : sigurinë, efikasitetin e largimit të defekteve , dhe mbështetjen e konsumatorit nuk janë përfshirë.
Mjetet do të përdoren për parashikuar dhe ruajtur informacionet konfidenciale dhe ndoshta të klasifikuara, të sigurisë është një kërkesë e ashpër , si dhe një numër të karakteristikave të sigurisë do të zbatohet, duke përfshirë edhe kriptimin e të gjitha informacioneve .

7.1 ANALIZA E KËRKESAVE NË VITIN 2049

Hapi i parë në mbledhjen e kërkesave të vitit 2049 do të jetë për të dërguar një agjent inteligjent për nxjerrjen e gjithë informacionet e përshtatshëm në lidhje me vlerësimin e programeve dhe planifikimin e tyre nga Ueb-i.

Të gjithë artikujt teknike dhe informacionet e marketingut do të jenë të mbledhura dhe të analizuara për mjete të ngjashme të tilla si : Application Insight, Artemis Views, Checkpoint, COCOMO dhe klonet e tij, Knowledge Plan, Microsoft Project, Price-S, SEER, SLIM, SPQR/20, SoftCost, dhe të gjitha mjetet e tjera të tilla.

Agjent inteligjent do të prodhojë një listë të konsoliduar të gjitha funksionet aktuale në dispozicion në të gjitha mjetet e ngjashme si : metodat e matjes , konvertimit të monedhës, rregullimet e normës së inflacionit, parashikimet e cilësisë, kostos totale të pronësisë, e kështu me radhë.

Shpresojmë, nga 2049ripërdorimi i softuerëve do të ketë arritur një nivel pjekurie në mënyrë që të ofroj katalogë të plotë të objekteve të ripërdorura që do të jenë në dispozicion, certifikimi për cilësi dhe siguri do të jene të zakonshme, arkitektura dhe projektimi do të kenë arritur në pikën ku përshkrimet standardet strukturore për aplikacione, pikat e lidhjes, dhe çështje të tjera të rëndësishme do të jetë e lehtë të arritshme. Agjent inteligjent do të mbledhin informata nga të dhënat publikimet në lidhje me numrin e kopjeve të mjeteve të tilla të shitura, të hyrat nga mjetet e shoqatave të përdoruesve për çështje gjyqësore kundër shitësit dhe tema të tjera të rëndësishme të biznesit .

Në qoftë se komponenti është përdorur për të vlerësuar aplikimet financiare të softuerëve agjentet inteligjent do të hetojnë në Ueb për të gjitha rregulloret e qeverisë që mund të jenë të aplikueshme të tilla si Sarbanes-Oxley dhe rregulla të tjera përkatëse. Për shkak të krizës financiare dhe recesionit rezultatet e rregulloreve të reja janë gati për publikim vetëm agjentet inteligjent dhe ekspert e sistemit mund t'i mbajnë . Për forma të tjera të softuerëve , agjentet inteligjent mund të kërkojnë në ueb për rregulloret, standardet, dhe tema të tjera që ndikojnë në qeverisje dhe në mandateve të qeverisë për shembull aplikacionet softuerike që merren me pajisjet mjekësore, të dhënat e procesit mjekësore, ose që kanë nevojë për privatësinë e mbrojtjes ligjore .

Faza e analizë se kërkesave do të shqyrtojë edhe platformat e mundshme për mjetet e reja të vlerësimit. Në çdo rast, një mjet i këtij lloji ndoshta do të kandidojë për platforma të shumëfishta dhe për këtë arsye duhet të jenë të planifikuara për Windows, Apple, Linux, Unix . Jo vetëm qe ky mjet do të veprojë mbi platforma të shumta, por edhe kjo është padyshim një vegël që do të jetë e vlefshme në shumë vende. Këtu edhe një agjent inteligjent do të vendoset për të kërkuar mjete të ngjashme të cilat janë në dispozicion në vende të tilla si Kina, Japonia, Rusia, Korea e Jugut, Brazil, Meksikë, dhe kështu me radhë. Ky informacion do të jetë pjesë e planit të planifikimit tregut dhe gjithashtu do të përdoret për të konstatuar se si shumë versione të duhet të ndërtohet me të dhëna të përkthyera në gjuhë të tjera natyrore. Duke përdorur informatat e mbledhura me anë të agjentëve të zgjuar në madhësinë e tregut aktual, një tjetër aspekt të analizë se kërkesave do të jetë për të parashikuar potencialet e tregut si një mjet të ri dhe karakteristikat e saj të ri në drejtim të konsumatorëve, të ardhurave, avantazhet konkurruese, e kështu me radhë.

7.2 DIZAJNIMI NË VITIN 2049

Për shkak se shumë aplikacione të ngjashme tashmë ekzistojnë, dhe për shkak se vetë kompania ka ndërtuar aplikacione të ngjashme, dizajnimi nuk fillon me një copë të pastër të letrës apo të një ekran të pastër. Në vend të kësaj dizajnimi fillon me një analizë të kujdesshme të arkitekturës dhe projektimit të aplikacionit . Një ndryshim shumë i rëndësishëm në dizajnim midis vitit 2009 dhe 2049 do të jetë përdorimi shumë karakteristik i standardit të ripërdorimit nga burimet shtëpiake , burimet komerciale apo ndoshta nga librarit e certifikuara për funksione të ripërdorshme .

Kompanitë e ndërtimit të aplikacioneve tashme kanë aplikacione të ngjashme, padyshim në shumë karakteristika të tilla si të vlerësimi të cilësisë, planifikimit të vlerësimit , dhe vlerësimit bazë të kostove qe do të jenë në dispozicion . Në mënyrë ideale, të paktën 85 për qind e karakteristikave dhe elemente të projektimit do të jetë në dispozicion në formë të ripërdorshme, dhe vetëm 15 për qind do të jetë me të vërtetë të reja dhe do të kërkojnë projektim te dizajnit . Për tiparet e reja, është e rëndësishme për të siguruar nivele të larta të cilësisë dhe të sigurisë, në mënyrë që inspektimet e projektimit do të kryhet në të gjitha karakteristika e reja që do të shtohen .

Prandaj, një ndryshim të madh rreth dizajnit ne vitin 2049 nga dizajni ne vitin 2009 është se po thuajse çdo funksion i ri do të jetë i projektuar si një objekti ripërdoruesh -em, në vend se duke u projektuar si një objekti unike për një aplikacion. Së bashku me ripërdorime formale si një qëllim të projektimit për të gjitha karakteristikat e rëndësishme të sigurisë, cilësisë, dhe transportueshmëris ndër platformat (Windows, Apple, Unix, Linux, etj) janë aspektet themelore të projektimit. Projektimi i Dizajnit për një aplikacion të vetëm duhet të eliminohet si një praktikë e përgjithshme, dhe të zëvendësohet me projektim te dizajnit për ripërdorimin që mbështet kërkesat e shumë aplikacioneve dhe shumë platformave .

Një sistem ekspert me një komponent të dizajnuar do të jetë i nevojshëm në mënyrë që të përfshijë karakteristikat e analizës statike, analiza komplekse, analiza të sigurisë, arkitekt-urës dhe projektimit, analiza strukturore, si dhe aftësinë e nxjerrjes se algoritmeve dhe rregullat e biznesit nga kodi i trashëguar . Inxhinierin softuerike për të bërë një disiplinë të vërtetë të inxhinierisë, ajo do të jetë e nevojshme që të ketë metoda efektive për analizimin dhe identifikimin optimale te dizajnimit të aplikacioneve softuerike . Një sistem ekspert që mund të analizojë strukturën, tiparet, përformance dhe përdorshmërin e aplikacioneve ekzistuese është një pjesë themelore e lëvizjes softuerëve nga një zanat në një disiplinë

inxhinierike. Gjithashtu, strukturat e standard arkitektonike janë të nevojshme dhe ndoshta mund të përcillni metodën e qasjes arkitektonike te Zachman [12].

7.3 ZHVILLIMI I SOFTUERIT NË VITIN 2049

Duke supozuar se ndoshta 85 për qind e karakteristika të aplikacioneve softuerike do të jenë në formën e komponentëve standarde të ripërdorshme, krijimi i programeve kompjuterike ne vitin 2049 do të jenë mjaft të ndryshme nga kodi i aplikacioneve qe zhvillohet sot rresht për rresht . Faza e parë e zhvillimit të programeve rreth 2049 është që të grumbullohen të gjitha komponentët ekzistuese të ripërdorshme dhe vënien e tyre së bashku në një prototip ku edhe zhvillimet e reja mund te shtohen me vone .

Ky prototip mund të përdoret për të vlerësuar çështje themelore të tilla si përdorshmeris , punën, sigurinë, cilësinë. Si tipare të reja që janë krijuar dhe testuar , ata mund të jenë bashkëngjitur në prototipin fillestare të punës. Kjo qasje është disi e ngjashme me zhvillimin e shkathët (agile development) , përveç se në shumicën e rasteve nuk do të fillojë me gjurmimin e të dhënat të aplikacioneve të trashëguara.

Team Software Process (TSP) dhe Personal Software Process (PSP) qasjet kanë treguar nivele shumë të larta të kontrollit të cilësisë Për shkak të sigurisë shumë të lartë dhe kërkesave cilësore për aplikacionet e reja kompanitë duhet qe komponentët e ripërdorshme duhet te sigurojnë qe defektet te jene afër nivelit zero. Përveç kësaj, historia e të gjithë komponentëve të ripërdorshme duhet mbledhur dhe analizuar për të vlerësuar dhe matur cilësinë dhe të sigurinë që mund të kenë qenë raportuar më parë.

7.4 DOKUMENTIMI NE VITIN 2049

Në 2009 mbështetja e klientëve dhe dokumentacioni i përdoruesit janë lidhje të dobëta për aplikacionet softuerike , dhe zakonisht renditen në mes të "papranueshme" dhe "margjinaleve." Disa kompani të tilla si Apple, IBM, dhe Lenovo herë pas here arrijnë nivele të "mirë", por jo shumë shpesh .

Ne aplikacionet të ndërtuara nga komponentë ripërdorshme do të ketë tekst ndihmës dhe informacione të përdoruesit si pjesë e paketës hapi i parë është që të mblidhen të gjitha seksionet e dokumentit të komponentët e ripërdorshme dhe të planifikohet për aplikim të

[12] http://en.wikipedia.org/wiki/Zachman_Framework

ri. Fillimisht do te shikonin dokumentacionin e përdoruesit dhe tekstin ndihmës hapi tjetër do të ishte për të dërguar një agjent inteligjent për të kontrolluar komente të përdoruesve të gjitha manualeve të konsumatorëve, . Natyrisht, si lavdërimet dhe ankesat në lidhje me këto tema janë të shumta në forumet edhe grupet e diskutimit por një agjent inteligjent do të jetë e nevojshme për të mbledhur ne një pasqyrë te plot . Shqyrtimet e librave të palës së tretë në ueb faqet të tilla si Amazon[13] gjithashtu do të analizoheshin .

Pasi agjent inteligjent ka përfunduar mbledhjen e të dhënave, shembuj të librave dhe tekstet e komentuara dhe të favorshme nga konsumatorët do të analizohen, duke përdorur komponent të automatizuar të tilla si FOG dhe Fleisch indexes . Qëllimi i këtij ushtrimi është për të gjetur strukturën dhe modelet e librave dhe të dhënave të përdoruesit që ofron informata më të mirë bazuar në vlerësimet e aplikacioneve të ngjashme nga konsumatorët . Pasi që dokumentet e shkëlqyera janë identifikuar, mund të jetë një ide e mirë për të nënkontraktuar punën në prodhimin e informacioneve të përdoruesit për autorët librat e të cilëve i kanë marrë përshtypjet më të mira për aplikacione të ngjashme . Në qoftë se këta autorë nuk janë në dispozicion, atëherë të paktën librat e tyre mund të sigurohet për autorët që janë në dispozicion dhe i cili do të krijojnë manual për përdoruesit.

Qëllimi është që të krijojë një model solid dhe të suksesshëm për ta ndjekur për të gjitha publikimet . Vini re se shkeljet e të drejtave të autorit nuk janë menduar Kjo është struktura e përgjithshme dhe renditja e informacionit që është e rëndësishme. Prandaj, pasqyrë e njeriut ndoshta ende do të luajnë një rol të madh në zhvillimin e materialeve të trajnimit. Dokumentacionet dhe materialeve të trajnimit do të duhet të jenë të përkthyera në disa gjuhë kombëtare, duke përdorur përkthimet e automatizuar si pikëni-sje . Shpresojmë, në 2049 përkthimin e automatizuar do të rezultojë në tekstin e duhur, dhe më shumë idioma se përkthimet e vitit 2011 .

7.5 MBESHTETJA E KLIENTIT NE VITIN 2049

Mbështetja e konsumatorëve, që aktualisht është edhe më keq se informatat e përdoruesit .

Problemet kryesore me mbështetjen e klientëve përfshijnë por nuk janë të kufizuara vetëm ne këto :

- ✓ Pritje të gjata duke u përpjekur për të arritur mbështetjen e klientëve me ane te telefonit

[13] http:// www.amazon.com

- ✓ Mbështetje të kufizuar telefonike për konsumatorët e shurdhër ose gjysmë i shurdhër
- ✓ Personeli dobët i aftësuar të cilët nuk mund të zgjidhin shumë pyetje
- ✓ Orë e kufizuar për mbështetjen e klientëve, që është orët e punës për një zonë tjetër .
- ✓ Përgjigjet të ngadalshme për pyetje e parashtruara ne e-mail .
- ✓ Akuzat për mbështetje të konsumatorëve edhe në raportet e gabimeve në softuerët e shitjes .
- ✓ Mungesa e analizave të gabimeve të njohura shpesh ose defekte .
- ✓ Mungesa e analizave për pyetjet më të shpeshta dhe përgjigjet .

Disa nga këto çështje janë shkaktare që programet janë duke u lirua në mënyrë rutinore me probleme (bugs) shumë të rënda, ose defektet që rreth 75 për qind e shërbimit të konsumatorëve bën thirrje për vitin e parë të përdorimit të aplikacionit rreth defekteve dhe problemeve. Kur softuerët të zhvillohet nga komponentët e certifikuar të ripërdorshme, dhe kur zhvillimi i ri synon të përafroj pranë nivelit zero defektet rreth vitit 2049 duhet të jenë ulur defektet të paktën 65 për qind në krahasim me normën e vitit 2009 . Kjo duhet të ndihmojë në afatin kohor të përgjigjeve me telefon dhe e-mail të pyetje të konsumatorëve. Çështja tjetër është mbështetje e pamjaftueshme për klientët e shurdhër dhe ata qe kanë vështirësi në dëgjimit . Këto çështje duhet të merren me shumë në konsideratë dhe të punojë më shumë shitësit e softuerit Përkthimi automatik i zërit në tekst duhet të jetë në dispozicion duke përdorur teknologjitë që i ngjajnë Dragon Naturally Speaking ose përkthyes të tjerë me zë, por shpresojmë se do të ketë përmirësuar në shpejtësi dhe saktësi në vitin 2049. Ndërsa pajisjet TTY dhe kompanitë e telefonit mund të ofrojnë ndihmë për të shurdhëritë dhe ata që kanë vështirësi në dëgjim, këto qasje janë të papërshtatshëm për probleme që kanë të bëjnë me raportet e softuerit dhe shërbimet ndaj klientit. Në mënyrë ideale për telefonat celularë dhe fiks mund të kenë një kombinim të veçantë të rëndësishme që tregon përdorimin nga një person i shurdhër ose gjysmë i shurdhër kur do të paraqite përkthimin automatik i zërit në tekst të ekranit mund të ofrohen nga shitësit apo ndoshta mund të vihet në dispozicion edhe nga prodhuesit e telefonave celular. Duhet të përdoret. metodat e zhvillimit të tilla si TSP dhe PSP, analiza statike, dhe inspektimet që të përmirësojnë cilësinë është teknikisht e mundur për të ndërtuar një sistem të ekspertëve për mbështje të klientëve e që përfshin identifimin e zërit, përkthimin e zërave në teksteve dhe teksteve në zëra , dhe një motor të inteligjencës artificiale që mund të flasin për konsumatorët, të dëgjoj për problemet e tyre, ndeshjen e problemeve kundër raporteve të tjera të sigurojë status të konsumatorit edhe për raste special ose të veçanta. Një kombinim i materialeve me cilësi së lartë të ripërdorshme dhe mbështetjen e sistemeve të ekspertëve për të analizuar defektet e softuerëve mund të bëjë përmirësime të rëndësishme në mbështetjen e klientëve .

7.6 MIRËMBAJTJA DHE PËRMIRËSIMET NË VITIN 2049

Pritshmëria mesatare e aplikacioneve softuerike shkojnë nga 10 në më shumë se 30 vjet , një proces i zhvillimi në vetvete nuk është i mjaftueshme për një disiplinë inxhinierike të vërtetë. Ajo është gjithashtu e nevojshme që të përfshijë në mirëmbajtje (riparimet defekt) dhe zgjerime (tipare të reja) për tërë jetën e aplikacioneve . Në fakt, jetëgjatësia maksimale për jetën e aplikacioneve të madhe është aktualisht e panjohur, sepse shumë prej tyre janë ende në shërbim. Në disa aplikacione , të tilla si për kontrollin e trafikut ajror, mund të themi përfundimisht se qe 50 vite po ofrojnë shërbim të vazhdueshëm. Aksidentalisht norma e rritjes së aplikacioneve softuerike pas vendosjes së tyre fillestare është rreth 8 për qind për një vit kalendarik, kështu që pas 20 deri 30 vjet të shfrytëzimit, aplikacionet janë fry në më shumë se dy herë në madhësinë e tyre origjinale .

Për fat të keq, kjo rritje është e shoqëruar zakonisht me rritje serioze në kompleksitetin e kushtëzuar dhe thelbësore, rezultatet e mirëmbajtjes bëhen progresivisht më të shtrenjta . Për të ngadalësuar prishjen dhe plakjen e aplikacioneve të trashëguara , ato duhet të rinovohen pas 5 apo 7 viteve të shërbimit . Renovimi do të eliminojë module të gabueshme të thjeshtojë kompleksitetin e kodit, të eliminoj të metat e sigurisë , dhe ndoshta edhe të konvertohet në kod të gjuhëve më moderne. Ajo duhet të jetë e qartë se mirëmbajtja e aplikacioneve softuerike të cilat janë ndërtuar pothuajse plotësisht nga komponentët e ripërdorshme që rrjedhin nga një numër burimesh do të jetë më të komplikuar për mirëmbajtje se aplikacionet në vitin 2009. Prandaj, është e nevojshme që të ketë informacion të saktë mbi burimet e çdo funksion në aplikacion. Kur gabim të ndodhë, duhet të jetë i njoftuar burimi origjinal Nëse defekti është nga një aplikacion ekzistuese pronarët dhe ekipet e mirëmbajtjes së aplikacion duhet të njoftohen. Për shkak se aplikacioni vepron në platforma të ndryshme (Windows, Apple, Linux, Unix, etj), ekziston edhe një shans i mirë që një defekt kur të raportohet në një platformë mund të jetë i pranishëm në versionet që veprojnë në edhe në platformat e tjera. Prandaj, një lloj çelësi i analizës do të përfshijë drejtimin e mjeteve statike dhe dinamike të analizave për çdo version sa herë që një defekt është raportuar .Në mirëmbajtjen ose riparimin e defekteve ne vitin 2049 duhet të ketë qasje në një mjedis pune të fuqishëm që integron bag-at dhe i raporton , analiza të automatizuar statike dhe dinamike, dhe mjetet e analizës së kompleksitetit .

Historikisht aplikacionet softuerike kanë tendencë të rriten me rreth 8 për qind për vit kalendarik duke përdorur madhësinë e lëshimit fillestar . Nuk ka asnjë arsye për të menduar se rritja në vitin 2049 do të jetë më e ngadalshëm se sa në vitin 2009, por ka disa arsye të mendoj se kjo mund të jetë më e shpejtë. Në 2049, një kombinim i agjentëve inteligjent edhe të sistemeve të ekspertëve do të vazhdojë në projektimit aktual për aq kohë sa kërkesa është shfrytëzuar. Të njëjtat lloje të sistemeve të ekspertëve që janë përdorur për rregullat e biznesit dhe algoritmet mund të mbahen në përdorim të vazhdueshëm për të siguruar se softuerët dhe materialet mbështetëse të saj janë gjithmonë në të njëjtat nivele të tërësisë. Kjo sjell deri në atë pikë që standardet për produktivitetin dhe cilësinë eventualisht mund të përfshijë më shumë se 30 dhe ndoshta edhe më shumë se 50 vjet . Prandaj, paraqitja e të dhënave ne standarde të tilla si ISBSG do të jetë një aktivitet i vazhdueshëm dhe jo një ngjarje e një kohë të caktuar

8. ENTERPRISE ARKITEKTURA DHE PORTFOLIO

Analiza ne vitin 2049, agjentët inteligjent, komponentët e projektuara nga ekspertët , dhe ekspert e mirëmbajtje mjediset e punës të bëhen të dislokuar gjerësisht, do të hapën forma të reja të punës që kanë të bëjnë me nivele më të larta të pronësisë së softuer-ve në nivel të ndërmarrjes dhe portfoliove. Në vitin 2009, korporatat dhe agjencitë qeveritare e mijëra e aplikacione softuerike janë zhvilluar gjatë shumë viteve dhe përdorimi i qasjeve të ndryshme arkitektonike, metodat e dizajnimit , metodat e zhvillimit, dhe gjuhët e program-imit. Përveç kësaj, shumë aplikacione në portofolio mund të jetë paketa komerciale të tilla si : paketa ERP, aplikacione të zyreve , aplikacione financiare, dhe të ngjashme. Këto aplikacione janë mbajtur në intervale të rastit. Shumica përmbajnë sasi të konsiderue-shme të metave të fshehura. Disa madje përmbajnë "module të gabueshme", të cilat janë shumë komplekse dhe me shumë gabime ne segment të kodit . Qëllimi i këtij ushtrimi është të identifikojë të meta e cilësisë dhe sigurisë në të gjitha aplikacionet e tanishme. Një karakteristikë tjetër që ka nevojë për analiza të ekspertëve dhe agjentëve inteligjente është qe të identifikohen pjesë e programeve që mund të kenë nevojë për ndryshime të reja për shkak të ndryshimeve në rregulloret e qeverive të ndryshme dhe ligjeve të tilla si : ndryshimet në ligjet tatimore, ndryshimet në politikat e qeverisjes, ndryshime në kërkesat e jetës private, dhe nevojat e të tjerët.

Vështirë se shpëton një ditë pa ndonjë ndryshim në ligjet ose rregulloreve shtetërore ose federale dhe, kështu që vetëm një kombinim i agjentëve inteligjent dhe të sistemeve të ekspertëve do të ishte në dijeni të asaj se çka mund të jetë e nevojshëm në një portofoli të qindra aplikacioneve . Që nga portofoliot e madhe mund të përfshijë më shumë se 10.000 kërkesa dhe 10 milion pika funksionale në tërësinë e tyre, kjo punë nuk mund lehtë të bëhet nga qeniet njerëzore dhe kërkon automatizim . Pak kompani në fakt e dinë madhësinë e portofoliove të tyre në drejtim të pikave funksionale apo rreshtave të kodit. Pak kompani në fakt e di koston e tyre të mirëmbajtjes dështimeve e , përmirësimeve dhe llojet tjera të punës. Pak kompani e din nivelet aktuale të cilësisë dhe të metat e sigurisë në softuerit ekzistues.

Nga 2049 është e mundur të parashikojnë një komplot të agjentëve inteligjente dhe sistemeve të eksperteve ku vazhdimisht do të punojnë në identifikimin e të metave ne seksionet e aplikacioneve të trashëguara e që kanë nevojë për vëmendje të duhur të cilësisë dhe sigurisë . Agjentët qe do të jenë gjeografikisht të shpërndara në mesin e 50 korporatave të ndryshme zhvilluese dhe lokacioneve mirëmbajtëse .Megjithatë rezultatet e këtyre mjeteve do të jenë të konsoliduara në nivelin e ndërmarrjes .Është e qartë se ka nevojë që softuerët të migrojnë nga një zanat që ndërton aplikacione rresht pas rreshti për një disiplinë inxhinierike që mund të ndërtojnë me cilësi të lartë sigurisë dhe cilësisë aplikacione nga komponentë të standardizuara. Disa inteligjente të sistemeve eksperte, metodave arkitektonike janë të nevojshme për ta arritur këtë . Përveç kësaj, metoda e analizës së automatizuar të sigurisë dhe analizës të cilësisë duke përdorur të dy këto analiza statike dhe dinamike duhet të jetë në përdorim të vazhdueshëm për të mbajtur aplikacionet të sigurta dhe të besueshëm.

Disa nga qëllimet e biznesit për këtë lloj të automatizuar të analizës portofolitos do të përfshijë qeverisjen e korporatave, bashkimet dhe blerjet, duke vlerësuar vlerat e tatueshme të aseteve softuerike, planifikimin e mirëmbajtjes, kontestet shkeljet e kontratës, dhe sigurisht përmirësimin e cilësisë dhe sigurisë .

Analiza e portfolios është veçanërisht e rëndësishme në rastin e bashkimit dhe blerjeve në mes të korporatave të mëdha. Përpjekje për të bashkuar portofoliot dhe organizatat e softuerike të dy kompanive të mëdha është një detyrë e frikshme që shpesh dëmton të dy partneret. Në nivel të enterprise arkitekturës dhe analizave të portofolios, përfaqësimet grafike do të jenë të vlefshme për të treguar përdorimin e softuerëve dhe statusin e të gjithë ndërmarrjes. Një aftësi të ngjashme me atë sot përdoret për Google Earth , fillojë me një pamje të nivelit të lartë të korporatës dhe të gjithë portofolios , dhe pastaj të nivelet tjera deri në nivelin e kërkesave individuale, njësive të veçanta të biznesit,dhe ndoshta edhe funksionet individuale dhe përdoruesit. Dallimi kryesor në mes të Google Earth dhe një përfaqësim të përgjithshëm të një portfolio të korporatave është se portofoli do të tregohet duke përdorur informacion te animuara dhe në kohë reale. Ideja është që të ketë përfaqësim të vazhdueshëm të animuar të rrjedhjes së informacionit të biznesit .

9. PËRFUNDIM

Ne fillim e cekem se Inxhinjeris Softuerike përfshin një fushë të gjere e cila ofrohet ne studimet themelore, master ,doktoraturës apo edhe me tutje .

Ne ketë punim jemi zhytur ne ketë fushë te madhe dhe kemi trajtuar disa tema te cilat gjenden rrallë te trajtuara neper literaturat qe ofrohen sot ne fushën e inxhinierisë softuerike.

Ne shumicën e kapitujve të trajtuar kemi ofruar informata më te shumta qe i hynë në pune me shumë ne fushën e menaxhimit të projekteve ku problemet qe i kemi trajtuar shumica prej tyre janë probleme që menaxheret e projekteve duhet qe t'i dine si t'i trajtojnë dhe te përdorin metodat e trajtimit me te avancuara te cilat ofrohen për menaxhimin e projekteve.

Arritjet ne lëmin e Inxhinierisë Softuerike janë te shumta përfshin shumë drejtime mirëpo nuk janë te mjaftueshme . Çdo dite shtohen softuer te ri shtohen gjuhe programuese të reja ide të reja mirëpo nuk kemi ndonjë standard të përgjithshëm qe duhet ti përmbahemi. Komponentët qe sot ofrohen janë krijuar nga vetëm nga një individ , grup apo kompani e caktuar . Ne kapitullin Zhvillimi i Softuerit ne vitin 2049 kemi shpalosur informata të shumta në pika të ndryshme se si do te rrjedh zhvillimi i softuerit ne vitin 2049 . Është një nisme e mirë qe te kemi disa standarde ku komponentët që do të zhvillohen të jenë të ripërdorshme . Zhvillimi i këtyre komponentëve do te krijoj mundësin e zhvillimit më të shpejt të softuerëve do te rris përformancen , sigurinë besueshmërinë . Këto komponentët do te ndikojnë ne zvogëlimin e kostove rritjen e produkteitivetit , cilësisë , sigurisë, aplikueshmeris të cilat do te zhvillohen dhe do të jene të shpërndara në të gjitha pikat e zhvillimit të softuerit.

Për fund neve te gjithëve qe merremi me fushën e inxhinierisë te softuerit na mbetet qe te studiojmë dhe ti fusim ne aplikim metodat e reja te zhvillimit te softuerit, arkitekturat , gjuhet e reja , komponentët dhe të punojmë ne avancimin e tyre ashtu siç planifikohet deri sa te arrijmë ripërdorshmeri te komponentëve dhe standarde te larta.

LISTA E FIGURAVE

LISTA E TABELAVE

AKRONIMET

ISBSG - International Software Benchmarking Standards Group
TSP - Team Software Process
PSP Personal Software Process
CEO - Chief Executive Officer
CTO - chief technology officer
CIO - chief information officer
AEA - Association of Enterprise Architects
AOGEA - Association of Open Group Enterprise Architects
GEAO - Global Enterprise Architecture Organization
IASA - International Association of Software Architects
WWIS - World Wide Institute of Software Architects
JEA - Journal of Enterprise Architecture
ROM - Read Only Memory
MIS - Management Information System
ERP - Enterprise Resource Planning
HTML - HyperText Markup Language
SQL - Structured Query Language
XML - Extensible Markup Language (XML)
CAD – Computer Aided Design
JAD - Joint Application Design
QFD - Quality Function Deployment
UML - Unified Modeling Language
RUP - Rational Unified Process
OMG - Management Object Group
SOA - Service-or (1)**ented architecture**
CIA - Central Intelligence Agency (1)

REFERENCAT

Librat :

1. **Evans, Eric.** *Domain-Driven Design: Tackling Complexity in the Heart of Software.* USA : Addison Wesley, 2003. 0-321-12521-5.

2. **Evans, Eric.**. *Domain-Driven Desgign Quickly.* USA : C4Media Inc., 2004. 978-1-4116-0925-9.

3. *Advances in Software Engineering.* **Domenik Slezak; Tai-hoon Kim; Akingbehin Kiumi ; Tao Jiang ; June Verner; Silvia Abrahao .** Germany : 2009, 2009. 1865-0929.

4. **JONES, CAPERS.** *Software Engineering Best Practices.* USA : The McGraw-Hill Companies, 2010. 978-0-07-162161-8.

5. *Advances in Software Engineering.* **Cisternino, Egon Börger and Antonio.** Island, Italy : Springer-Verlag Berlin Haidelberg , 2007. 2008.

Librarit online të cilat janë përdorur për kërkime :

1. Wiley InterScience, http://www3.interscience.wiley.com
2. UMI Proquest, http://proquest.umi.com/login
3. EIFL Direct / EBSCOhost, http://search.ebscohost.com
4. IEEE plore : http://ieeexplore.ieee.org
5. Springer : http://www.springerlink.com
6. Hindwai : http://www.hindawi.com/journals/ase/
7. Sc. Direct : http://www.sciencedirect.com/science/journal/09659978

Lidhje tjera :

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

http://en.wikipedia.org/wiki/Zachman_Framework

http://www.sans.org/

http://www.cwe-mitre.org

http://people.ku.edu

http://www.extremeprogramming.org/

http://agilemethodology.org/

http://www.agileadvice.com/archives/2006/01/the_pros_and_co.html

http://en.wikipedia.org/wiki/Waterfall_model