

PROPUNERE

**Servicii de mentenanță a Sistemului Informațional e-Rețetă
compensată**

Beneficiar: Compania Națională de Asigurări în Medicină

Septembrie 2025

Cuprins

Informații de contact ale Ofertantului	4
PROPUNEREA TEHNICĂ	5
1. Prezentarea companiei	6
1.1. Domenii de expertiză	6
1.1.1. Servicii de consultanță specializate	6
1.1.2. Servicii de dezvoltare	6
1.1.3. Integrare echipamente	7
1.1.4. Web development	7
1.2. Lista de tehnologii	7
2. Planul de implementare al proiectului	8
2.1. Infrastructură și securitate	8
2.2. Mentenanță corectivă	8
2.3. Mentenanță preventivă	8
2.4. Mentenanță adaptivă	8
3. Planul de comunicare	9
4. Controlul schimbărilor	10
5. Asigurarea calității și securității informației	10
6. Metodologie de analiză și dezvoltare software	11
6.1. Etapa de analiză	11
6.2. Etapa de proiectare	12
6.3. Etapa de dezvoltare	13
7. Metodologia de testare	16
7.1. Introducere	16
7.2. Tehnici de testare software	17
7.2.1. Testarea arhitecturii Client/Server	17
7.2.2. Test de compatibilitate și configurare	17
7.2.3. Testarea utilizabilității	17
7.3. Strategii de testare	18
7.3.1. Unit testing	18
7.3.2. Testul de integrare	19
7.3.3. Testarea de sus în jos	19
7.3.4. Testarea de jos în sus	20
7.3.5. Metoda mixtă	20
7.3.6. Testul de validitate	20
7.3.7. Testul de acceptare	21
7.3.8. Test de sistem	21
Testul de recuperare	21
Testul de securitate	21
Testul de stres	21
Testul de performanță	22
7.3.9. Testare regresivă	22
7.3.10. Testare software orientat obiect	23
7.3.11. Metoda de testare empirică	23
7.3.12. Metoda de testare ierarhică	24
7.4. Efectuarea testării	24

7.5. Indicatori de testare	25
8. Intrarea în producție	26
9. Instruirea utilizatorilor	26
10. Asistență tehnică și suport post-implementare	26
10.1 Clasificarea problemelor	27
10.2. Soluționarea problemelor la distanță	28
10.3. Serviciul timpurilor tratării problemelor	28
10.4. Orarul serviciilor	28
10.5. Raport de activitate pentru servicii de mentenanță și suport	29
11. Metodologie de analiză și combatere a riscurilor	29
Metodologie definire Riscuri	30
Riscurile proiectului și planul de tratare al acestora	32
Soluții de preîntâmpinare a riscurilor	34
12. Echipa de proiect	35
ANEXE	37
- CV-urile echipei de proiect	37

Informații de contact ale Ofertantului

Această propunere este elaborată de către Das Soft Plus S.R.L., o companie din Republica Moldova specializată în domeniul dezvoltării software.

Vă rugăm să atrageți atenția la datele de identificare de mai jos ale consultantului și datele persoanei de contact:

- **Datele companiei:**

Das Soft Plus S.R.L.

Număr de identificare de stat – cod fiscal: 1019600011052, cod TVA: 0210173

Adresa: MD-2001, str. Lev Tolstoi, 74, ap. 78, Chișinău, Republica Moldova

Rechizite bancare: IBAN: MD33VI022241400000287MDL, SWIFT: VICBMD2X446,
B.C."VICTORIABANK"S.A. suc.nr.14 Chișinău

Telefon: +373 69 393 169

Mob: +373 68 160 961

Email: dasoftplus@gmail.com

Website: <https://corlab.tech/>

- **Datele persoanei de contact:**

Dascal Mihai,

Manager Proiect

+373 68 160 961

Email: dascal.mi@gmail.com

Butucea Afanasie,

Administrator

+373 69 393 169

Email: dasoftplus@gmail.com

PROPUNEREA TEHNICĂ

Denumirea companiei :	Das Soft Plus S.R.L.
Țara de origine:	Republica Moldova
Reprezentantul companiei:	Butucea Afanasie
Adresa:	MD-2001, str. Lev Tolstoi, 74, ap. 78, Chișinău, Republica Moldova
Tel / Fax:	Tel: +373 69 393 169, Mob: +373 68 160 961
Email:	<u>dasoftplus@gmail.com</u>
Website:	<u>https://corlab.tech/</u>

1. Prezentarea companiei

Suntem o companie fondată în 2019, la inițiativa a doi dezvoltatori și un designer, extrem de motivați de provocările și satisfacțiile oferite de mediul antreprenorial. Clienții ne-au învățat cel mai mult despre dezvoltarea de software, deoarece ne-au motivat să găsim soluții compatibile cu nevoile lor. Orice problemă are o soluție, iar o soluție bună poate fi generalizată. De aceea, în următorii 5 ani, am dezvoltat un nou model de afaceri: înțelegând tot mai bine trendurile și oportunitățile de piață, am încercat să convertim fiecare proiect într-un produs eficient și maxim scalabil. Ne-am concentrat pe cercetarea diferitelor modele de afaceri pentru a identifica acele puncte importante care ar ajuta cu adevărat administratorii unei afaceri, în funcție de industria din care fac parte.

Suntem ferm convinși că am reușit să creăm atât soluții generice, cât și soluții personalizate, pornind de la cerințele specifice ale clienților și continuând cu dezvoltarea ulterioară la nevoile acestora. În prezent, portofoliul companiei include soluții pentru următoarele industrii: medicină, biotehnologie, învățarea automată și logistică, folosind cele mai noi tehnologii.

Putem afirma că avem o vastă experiență în dezvoltarea și implementarea de sisteme informatice complexe, în special în industria asistenței medicale, deoarece unul dintre fondatorii companiei are studii superioare și experiență în medicină și depune toată pasiunea pentru a crea soluții eficiente. Compania a cunoscut o creștere continuă, iar în prezent, echipa noastră are peste 18 membri și acoperim piețele din SUA și Europa, în special țările vorbitoare de limbă engleză.

Pe lângă portofoliul nostru diversificat de clienți, suntem bucuroși să anunțăm că lucrăm la un important proiect **guvernamental** pentru **CNAM**. Proiectul nostru comun, numit e-Rețetă, are ca obiectiv crearea unei platforme digitale integrate pentru gestionarea și procesarea rețetelor medicale în mod electronic.

De asemenea, proiectul este integrat cu **serviciile guvernamentale**, ceea ce ne oferă experiența necesară pentru a livra cu succes proiectul. Echipa noastră a depus efortul necesar pentru a finaliza proiectul în termeni și bugetul alocat, iar rezultatele obținute până acum sunt promițătoare.

1.1. Domenii de expertiză

Dezvoltatorul este o companie ce acoperă o arie largă de activități legate de domeniul informatic și al tehnicii de calcul, de la servicii de consultanță specializate, dezvoltare și analiză software, instalări și configurări de rețele până la integrări de sisteme informatice și soluții software la cheie.

1.1.1. Servicii de consultanță specializate

- Analiza și definire cerințe utilizator;
- Achiziții hardware și software;
- Implementare soluții ERP/ CRM;
- Creare soluții de management al documentelor și ale fluxurilor de lucru;
- Scriere specificații tehnice;
- Estimare costuri și management proiecte IT;

1.1.2. Servicii de dezvoltare

Deși competența noastră de bază este pe tehnologii Microsoft, acoperim întreaga paletă de soluții pentru proiectarea, administrarea și optimizarea bazelor de date SQL și NoSQL, după cum urmează:

- Aplicații backend, web, desktop, mobile: documentare, dezvoltare, portare;
- Proiectare și modelare baze de date relaționale;
- Proiectare și modelare baze de date non-relaționale;
- Optimizare baze de date;
- Portare aplicații de pe diverse platforme sau diverse limbaje;
- Proiectare interfețe utilizator;
- Programare multiplatforma;

1.1.3. Integrare echipamente

Echipamente hardware: instalare, configurare, integrare în aplicații, programare embedded și testări funcționale.

- Instalare echipamente;
- Configurare echipamente;
- Integrare echipamente în aplicații;
- Testare funcțională pe unitate și în sistem - scenarii de test;

1.1.4. Web development

Site-uri de prezentare sau magazine online de la A la Z: web-design, dezvoltare, scriere de conținut, consultanță SEO.

1.2. Lista de tehnologii

Web și Aplicații → Angular, React.js, Next.js, Vue.js, Go Lang, Node.js, Nest.js, Express.js

Mobile → ReactNative, Ionic

Baze de date → ElasticSearch, MongoDB, MySQL, neo4j, PSQL

Cloud și Arhitectură → AWS, Azure, DigitalOcean, Docker Compose, Google Cloud, Kubernetes

Testare și asigurarea calității → Python, Java, K6, Selenium, Appium, Jmeter, Taiko

UX/UI Design → AdobeXD, Figma, Framer, Illustrator, Invision, Photoshop

2. Planul de implementare al proiectului

Ne angajăm să asigurăm servicii de mentenanță corectivă, preventivă, adaptivă și suport tehnic pentru Sistemul Informațional Automatizat „eRețeta compensată”, conform cerințelor din caietul de sarcini, după cum urmează:

2.1. Infrastructură și securitate

- Administrare pe platforma MCloud guvernamentală, cu orchestrare prin Kubernetes și management prin Rancher.
- Servere aplicații dezvoltate pe NestJS/TypeScript, front-end ReactJS, proxy NGINX.
- Baze de date: PostgreSQL (date structurate), MongoDB (rapoarte și loguri), Redis (cache).
- Integrare și interoperabilitate cu servicii guvernamentale și externe: MPass, MSign, MLog, AOAM, MCloud API Gateway.
- Disponibilitate minimă 99,9% (SLA), monitorizare prin Elastic APM, Kibana, Elasticsearch.
- Implementarea de măsuri avansate de securitate: VPN securizat, JWT, politici de backup și testare restaurare, teste de penetrare periodice.

2.2. Mentenanță corectivă

- Investigarea și remedierea incidentelor și erorilor raportate prin Service Desk (ticketing).
- Corectarea bug-urilor la nivel de aplicație, baze de date și API-uri.
- Documentarea tuturor modificărilor și actualizarea codului sursă și a rapoartelor tehnice.

2.3. Mentenanță preventivă

- Verificarea periodică a serverelor, microserviciilor și bazelor de date.
- Monitorizarea performanței, spațiului pe disc și securității infrastructurii.
- Optimizarea codului sursă și eliminarea vulnerabilităților.
- Instalarea patch-urilor și actualizărilor de securitate pentru OS și middleware.
- Teste de penetrare pentru identificarea și prevenirea atacurilor cibernetice.

2.4. Mentenanță adaptivă

- Asigurarea compatibilității cu noile versiuni ale serviciilor externe (ex.: MPass, MSign, AOAM).
- Adaptarea rapoartelor și șabloanelor existente conform cerințelor CNAM.
- Modificarea registrelor și structurii bazei de date la introducerea de noi clasificatoare medicale.

- Implementarea de noi API-uri pentru interoperabilitate cu sisteme externe.
- Ajustarea serviciilor în funcție de reglementările legale și schimbările de politici publice.
- Suport tehnic pentru utilizatori prin **Hot-Line, email, remote access, intervenții on-site**.

3. Planul de comunicare

Comunicarea între reprezentanții noștri și reprezentanții beneficiarului se realizează pe nivele organizaționale, în funcție de tematica fiecărei comunicări, astfel:

Comunicarea oficială între părți se derulează prin mijloace scrise sau verbale, la nivel de Manager de Proiect asignat din partea societății noastre – Manager de Proiect (responsabil de contract) din partea beneficiarului.

În cadrul primei ședințe de proiect (kick-off meeting), vor fi stabilite canalele și mijloacele de comunicare între părți (persoane de contact, numere de telefon, fax, adrese de e-mail).

În vederea eficientizării comunicării între părți și reducerii timpului necesar implementării proiectului, aspecte tehnice și de implementare pot fi comunicate între experți tehnici din partea prestatorului și beneficiarului, cu menținerea managerilor de proiect în *carbon-copy*.

Managerul de proiect are rolul de a asigura faptul că deciziile luate cu referire la implementarea proiectului sunt aduse la cunoștința echipei de implementare din subordine și sunt puse în practică. Managerul de proiect se asigură că echipa de proiect este informată asupra documentelor de analiză și schimbărilor ce apar, iar fiecare persoană își cunoaște rolul în cadrul proiectului și acțiunile pe care trebuie să le desfășoare.

Comunicarea se realizează prin ședințe de lansare a etapelor, informări periodice și prin contactul informațional informal specific relațiilor de lucru în echipă.

Managerul de proiect înaintează ori de câte ori este necesar și la finalul fiecărei etape (sprint), informări adresate beneficiarului cu privire la stadiul proiectului și starea etapei, probleme și necesarul de schimbări, riscuri identificate/materializate și modalități de management al acestora, rapoarte specifice.

Comunicarea în cadrul proiectului se realizează pe două paliere: comunicarea în cadrul echipei proprii a prestatorului și comunicarea cu beneficiarul.

Comunicarea internă în cadrul societății noastre se realizează prin mijloace formale și informale. Mijloacele formale includ ședințe periodice conform metodologiei agile, precum și utilizarea unei platforme proprii de management al proiectelor. Platforma utilizată presupune înregistrarea membrilor echipei, înregistrarea sarcinilor de lucru (task-uri), definirea atribuțiilor pentru fiecare membru al echipei, stabilirea termenelor intermediare și finale de realizare a atribuțiilor, stabilirea ședințelor de lucru periodice și înregistrarea lor ca activități, urmărirea periodică a punctelor de reper, ședințe de progres, actualizarea permanentă a graficului de implementare a proiectului, echipei implicate, sarcinilor fiecărui membru al echipei și alte aspecte organizatorice.

Partajarea documentelor aferente implementării proiectului între membrii echipei se realizează prin intermediul unor servicii de versionare, astfel încât modificările apărute asupra documentelor să fie ușor de urmărit.

Comunicarea cu reprezentanții beneficiarului se realizează diferențiat în funcție de tipul și rolul informațiilor comunicate, astfel:

- Informația cu referire la modul de implementare al proiectului, livrabile, raportări de etapă, schimbări de specificații, și orice alte comunicări ce pot afecta implementarea proiectului se realizează la nivel de manager de proiect

- Informația cu referire la detalii tehnice de implementare poate fi comunicată de către personalul tehnic și de implementare, cu informarea managerilor de proiect
- Informația critică și care necesită luarea unor decizii la nivel instituțional cu privire la implementarea proiectului va fi comunicată de către Managerul de Proiect sau de către comitetul de conducere al proiectului, respectiv conducătorii celor două entități juridice.

4. Controlul schimbărilor

Comitetul de conducere al proiectului poate decide, în vederea soluționării unei probleme sau în urma identificării unui curs mai bun de acțiune, efectuarea de schimbări ale planului de lucru stabilit în analiza inițială.

O schimbare în cadrul proiectului poate apărea în următoarele cazuri:

- Cerere de schimbare ale specificațiilor
- Schimbarea cerințelor utilizatorilor
- Sugestie de îmbunătățire a unuia sau mai multor livrabile din proiect
- Solicitarea din partea beneficiarului sau utilizatorilor de adăugare a unui nou criteriu de acceptanță
- Modificări organizaționale care duc la necesitatea adaptării livrabilelor proiectului
- Imposibilitatea prestatorului de a livra tot ceea ce este contractat în limitele de cost și timp stabilite
- Neconformitate
- Clarificare
- Modificări legislative
- Inovație în tehnologie

Managerul de proiect va consemna în registrul de schimbări (back log în etapa de dezvoltare) fiecare solicitare de schimbare, cu indicarea priorității problemei, astfel:

- Obligatorie – livrabilul nu va funcționa fără modificarea respectivă
- Importantă – absența schimbării ar fi un inconvenient
- Utilă – schimbarea nu este vitală
- Cosmetică – fără importanță pentru funcționarea livrabilului
- Nu necesită schimbare

De asemenea, va fi consemnată analiza de impact, decizia luată cu privire la realizarea schimbării, precum și autorul deciziei. Pentru problemele care necesită schimbări, Managerul de proiect face o analiză pentru a identifica ce trebuie să se schimbe, ce efort presupune schimbarea, care este impactul asupra justificării economice a proiectului, care este impactul asupra riscurilor. Rezultatul acestei analize este o *Cerere de Schimbare*, care va conține data, numărul din registrul schimbărilor, starea, descrierea schimbării, impactul schimbării, evaluarea priorității, decizia, responsabilitatea pentru implementarea schimbării, data când cererea de schimbare a fost alocată în vederea implementării. Aceste informații sunt stocate în platforma utilizată pentru managementul proiectului / implementării.

Managerul de proiect nu autorizează fără aprobare din partea Comitetului de Conducere al Proiectului (conducerea instituției proprii și a beneficiarului) nici un fel de acțiune care ar conduce la modificarea unui livrabil care a fost deja aprobat / recepționa.

5. Asigurarea calității și securității informației

Dezvoltatorul utilizează o metodologie internă de asigurare a calității, folosind cel puțin două niveluri de revizuire a muncii, precum și a rezultatelor preconizate, care urmează să fie trimise la clienți, pentru a se asigura că livrabilele furnizate sunt de cea mai înaltă calitate. Astfel, fiecare raport sau livrabil este revizuit de o persoană cu abilități de revizuire și de un nivel superior de calificare.

În proiectele de dezvoltare software, metodologia noastră presupune că întregul cod sursă elaborat de echipa de dezvoltare să fie revizuit regulat de către liderul de echipă pentru a se îmbunătăți în permanență calitatea produsului livrat.

Scopul implementării SMSI este de gestionarea și controlarea mai bine riscurile de securitate a informațiilor și de a proteja confidențialitatea, integritatea și disponibilitatea informațiilor. Componentele SMSI includ politici, proceduri, planuri, roluri, responsabilități și resurse care au fost implementate pentru a gestiona riscurile de securitate a informațiilor și protejarea acestora.

6. Metodologie de analiză și dezvoltare software

Etapile procesului de dezvoltare și implementare a sistemului:

- Analiză
- Proiectare
- Dezvoltare
- Instruirea utilizatorilor
- Intrarea în producție
- Asistență tehnică și suport post implementare

6.1. Etapa de analiză

Etapa de analiză presupune determinarea cerințelor funcționale ale sistemului, pornind de la analiza de nevoi generată de activitatea beneficiarului și coroborând această informație cu datele despre infrastructura tehnică și de securitate, utilizatorii sistemului, cazurile de utilizare, fluxurile și procesele de lucru, reglementări legale și interne ale beneficiarului, procedurile de lucru specifice și standardele internaționale în domeniu.

Activitățile de analiză se derulează la sediul beneficiarului (pentru identificarea nevoilor specifice, proceselor de lucru și reglementărilor interne) și la sediul prestatorului pentru integrarea tuturor datelor, elaborarea documentului de analiză și analiza componentelor tehnice și a detaliilor de implementare.

Rezultatele etapei de analiză sunt reprezentate de un pachet de specificații funcționale agreat de comun acord cu beneficiarul.

Etapa de analiză este constituită din 4 sub-etape, astfel:

- Etapa de **analiză generală**, în cadrul căreia se realizează
 - Analiza situației curente în cadrul instituției beneficiarului
 - Stabilirea obiectivelor generale ale proiectului

- Stabilirea obiectivelor specifice prin rafinarea obiectivelor generale
- Stabilirea părților implicate (entități organizatorice din cadrul organizației Beneficiarului, implicate în utilizarea și administrarea produsului ce urmează a fi livrat)
- Stabilirea cerințelor funcționale pentru parte implicată
- Stabilirea arhitecturii generale – stabilirea componentelor/modulelor necesare
- Etapa de **analiză detaliată**, în cadrul căreia se realizează
 - Identificarea utilizatorilor sistemului
 - Identificarea și analiza infrastructurilor tehnice și de comunicații existente la beneficiar
 - Identificarea proceselor de lucru și a fluxurilor de activități corespunzătoare ariei de acoperire a proiectului
 - Detalierea fluxurilor în procese/activități corespunzătoare
 - Identificarea rolurilor și responsabilităților pentru fiecare activitate
 - Trasabilitate activităților identificate cu cerințele părților implicate identificate în cadrul fazei de analiza generala pentru a asigura acoperirea întregii arii de activitate solicitate
 - Identificarea cerințelor funcționale pe baza activităților
 - Trasabilitatea cerințelor funcționale cu funcționalitățile aplicației standard
 - Identificarea necesităților de modificare a funcționalităților standard și de dezvoltare a funcționalităților suplimentare
 - Identificarea problemelor și riscurilor potențiale ce pot afecta implementarea / utilizarea sistemului (inclusiv eventuale probleme de incompatibilitate între diverse module)
 - Identificarea măsurilor de reducere a riscurilor și remediere a eventualelor probleme ce pot fi prevăzute în această etapă
 - Stabilirea și definirea grupurilor de lucru
 - Identificarea necesităților de instruire pentru utilizatorii finali
 - Stabilirea planului de testare funcțională a produsului care va fi implementat
 - Elaborarea planului de mentenanță și a oportunităților de dezvoltări viitoare ale sistemului
- Etapa de **elaborare a raportului de analiză**
- **Livrarea și acceptarea raportului de analiză**
 - Raportul de analiză livrat și acceptat de către beneficiar reprezintă documentul de referință cu descrieri funcționale, în baza căruia se dezvoltă și livrează produsele.
 - Planul de testare funcțională și planul de acceptanță constituie anexe la raportul de analiză și vor reprezenta documentele în baza cărora vor fi verificate, validate și acceptate livrabilele în cadrul proiectului

6.2. Etapa de proiectare

Etapa de proiectare are rolul de a defini specificațiile tehnice ale produselor ce urmează a fi livrate. Această etapă are ca input raportul de analiză și anexele acestuia.

În etapa de proiectare este definită arhitectura funcțională a livrabilelor. Această activitate presupune elaborarea scenariilor de utilizare de detaliu, la nivel de cazuri de utilizare, inclusiv rolurile asociate fiecărui caz, constrângerile și regulile aferente, precum și eventualele excepții.

De asemenea, se realizează definirea arhitecturii tehnice, respectiv definirea sistemului din punct de vedere tehnic, cu definirea infrastructurii suport (servere, stații de lucru, infrastructuri și protocoale de comunicații, surse de date, stocarea datelor, etc)

Dezvoltarea modelului informațional presupune precizarea arhitecturii de date a sistemului, la nivel logic și fizic.

Modulele sistemului sunt proiectate și descrise în această etapă.

Proiectarea sistemului poate identifica mai multe soluții, urmărindu-se ușurința și eficiența realizării și implementării cerințelor beneficiarului, cu respectarea restricțiilor de ordin tehnic, organizatoric, financiar sau legal.

Procesul de proiectare pornește de la nevoile și prioritățile din organizația beneficiară, fiind esențială implicarea utilizatorilor sistemului, în scopul înțelegerii corecte a proceselor de lucru și acceptanța utilizatorilor cu privire la noul sistem.

6.3. Etapa de dezvoltare

Având în vedere evoluția rapidă a tehnologiilor, modificările legislative frecvente, precum și nevoia sistemelor informatice de a ține pasul cu aceste schimbări, utilizăm metodologii agile de dezvoltare a sistemelor informatice, astfel:

- Considerăm că indivizii și interacțiunea sunt mai importante decât procesele și instrumentele
- Considerăm că un software funcțional este mai important decât o documentație foarte amplă
- Considerăm prioritară colaborarea cu clientul față de negocierea contractului
- Receptivitatea la schimbare este mai importantă decât urmărirea unui plan

Astfel, produsele noastre sunt centrate în jurul omului, al beneficiarului și al utilizatorului final. Prioritatea noastră este reprezentată de satisfacerea nevoilor beneficiarului și livrarea la timp a produselor. Livrările de soft (module) funcționale se realizează periodic, cu preferință pentru termene scurte.

Principiul metodologiei noastre de lucru (SCRUM) constă în dezvoltarea incrementală a aplicațiilor software, implicând totodată păstrarea unei liste transparente cu cererile de modificare/schimbare (backlog).

Prin utilizarea metodologiei *agile* de dezvoltare software sunt reduse riscurile de dezvoltare și timpul de execuție prin implementarea proiectelor în formă foarte flexibilă și interactivă.

Procesul de lucru este unul iterativ și incremental, ce reprezintă planul unui proiect care include un set de activități și roluri predefinite. Principalele roluri sunt cele de *Conducător scrum* (*scrum master*) care întreține procesele și are rolul de manager de proiect; *Deținător de produs* – reprezintă vocea clientului, și *echipa* de dezvoltatori software.

Sarcinile sunt organizate în *sprint-uri*, iar definirea sprint-urilor se realizează în urma unor întâlniri de planificare sprint. Pe parcursul acestei întâlniri Deținătorul de proiect informează Echipa cu privire la sarcinile nerezolvate pe care dorește să le abordeze. Echipa stabilește câte astfel de sarcini poate îndeplini până la următorul sprint. În timpul unui sprint nu se pot schimba sarcinile alese. La sfârșit Echipa demonstrează cum se utilizează produsul intermediar obținut.

Acest mod de lucru se reflectă la modul de organizare al echipei interne. Echipele beneficiază de independență și auto-organizare, cu comunicare verbală între toți membrii echipei și între diferitele departamente care au legătură cu proiectul.

Este general acceptat faptul că pe parcursul dezvoltării unui proiect, beneficiarul se poate răzgândi cu privire la ce dorește de la produsul software. Astfel de schimbări sunt imprevizibile și nu sunt ușor de adaptat la proiect prin metode tradiționale de dezvoltare software.

Persoanele implicate în dezvoltarea produsului sunt împărțite în două categorii:

- cei direct implicați în procesul de dezvoltare, angajați să construiască proiectul și care sunt trași la răspundere.
 - **Conducătorul Scrum** – are un rol de manager de proiect (dar el nu este șeful echipei) ce trebuie să se asigure că procesul de dezvoltare evoluează în conformitate cu tehnicile, valorile și regulile Scrum. Acesta interacționează atât cu Echipa de dezvoltare, cât și cu clienții și conducerea organizației. Este de asemenea responsabil să se asigure că orice impediment și orice element care distrage atenția echipei sunt înlăturate, astfel încât productivitatea echipei să fie permanent la un nivel ridicat.
 - **Deținătorul de produs** – reprezintă vocea, interesele clientului. El este responsabil de proiectarea, administrarea, controlul și prezentarea produsului nerezolvat; ia decizia finală cu privire la sarcinile din produsului nerezolvat și le asociază priorități. Este ales de către Conducătorul Scrum, client și conducere.
 - **Echipa** – este responsabilă cu dezvoltarea produsului; are autoritatea de a decide ce măsuri trebuie luate pentru a rezolva sarcina asociată fiecărui sprint și are dreptul de a se auto-organiza tot în același scop. În general o echipă Scrum este alcătuită din 5-9 persoane.
- cei care nu sunt implicați direct în dezvoltarea proiectului, dar de a căror părere trebuie să se țină cont. În abordarea agilă un aspect foarte important îl reprezintă implicarea utilizatorilor, clienților, oamenilor de afaceri în procesul de dezvoltare. Aceștia trebuie să ofere feedback cu privire la rezultatele fiecărui sprint pentru a adapta și îmbunătăți viitoarele procese de lucru.
 - **Utilizatorii** – cei care vor folosi produsul software
 - **Clienții** – cei care stabilesc scopul proiectului; sunt implicați în procesul de dezvoltare doar când are loc evaluarea unui sprint
 - **Managerii** – cei responsabili de luarea deciziilor finale. Participă de asemenea în stabilirea obiectivelor și a condițiilor de lucru

Utilizăm metode și instrumente de management în diferite faze, pentru a evita confuzia creată de imprevizibilitatea și complexitatea proiectelor și schimbărilor. Elementele caracteristice sunt:

- **Sprint-ul** – perioadă de 15-30 zile (durata exactă este stabilită de către Echipă). El include faze tradiționale de dezvoltare software precum etape de cerințe, analiză, design, evoluție, livrare. Echipa Scrum se organizează astfel încât să producă o nouă unitate funcțională a produsului la sfârșitul unui sprint. Cu ajutorul acestora, sistemul se adaptează mai ușor la schimbări.
- **Produsul nerezolvat** – mulțime de sarcini nerezolvate: descrieri ale funcționalităților și serviciilor dorite – tot ce este nevoie pentru a atinge obiectivul final – așa cum este el definit în prezent; poate fi modificat de către oricine. Sarcinile au asociate priorități de către Deținătorul de produs în funcție de valoarea lor din punct de vedere al afacerii (valoare stabilită de către Deținătorul de produs), și de efortul necesar dezvoltării acestora (stabilit de către Echipă). Este actualizat în mod constant prin adăugare, modificare, specificare, înlăturare, stabilire de priorități cu privire la elementele conținute. Câteva dintre elementele ce pot face parte produsul nerezolvat sunt implementarea anumitor funcții, remedierea bug-urilor, înlăturarea defectelor, îmbunătățirea diferitelor componente. Printre cei care pot participa la construirea acestui produs sunt clienții, Echipa de dezvoltare, echipa de marketing și vânzări. Deținătorul de produs este cel responsabil cu administrarea produsului nerezolvat.
- **Sprint-ul nerezolvat** – reprezintă un document, detaliat, bazat pe elementele din produsul nerezolvat ce vor fi adresate în următorul sprint; conține informații despre modul în care Echipa va implementa cerințele stabilite. Sarcinile sunt împărțite pe ore astfel încât nici o sarcină să dureze mai mult de 16 ore (daca o sarcină ar dura mai mult, ea trebuie împărțită în sarcini mai mici). Sarcinile nu sunt repartizate angajaților – aceștia au libertatea de a-și alege sarcinile

dorite. Cei care stabilesc ce elemente vor face parte din sprint-ul nerezolvat sunt Conducătorul Scrum, Deținătorul de produs și Echipa în cadrul întâlnirii de planificare sprint pe baza priorităților și a obiectivelor stabilite pentru acel sprint. Sprint-ul nerezolvat este stabil până când sprint-ul este terminat. Când toate sarcinile din sprint-ul nerezolvat sunt împlinite o nouă iterație a sistemului este finalizată.

- **Burn down** - este un grafic afișat la vedere ce reprezintă timpul și munca rămasă până la terminarea proiectului. Este actualizat zilnic și ajută la estimarea datei la care va gata produsul.
- **Estimarea efortului** – este proces iterativ în care se concentrează atenția spre estimarea cât mai precisă a efortului depus pentru tratarea unei sarcini nerezolvate atunci când există mai multe informații despre acea sarcină.
- **Întâlnirea de planificare a unui sprint** – reprezintă o întâlnire în două etape organizată de Conducătorul Scrum. La prima parte a întâlnirii iau parte clienții, utilizatorii, conducerea, Deținătorul de produs și Echipa, pentru a decide obiectivele și funcționalitatea următorului sprint. La a doua parte a întâlnirii participă Conducătorul Scrum și Echipa pentru a discuta despre cum se va implementa unitatea produsului reprezentată de acest sprint.
- **Întâlnirea Scrum zilnică** – este organizată pentru a urmări continuu progresul echipei; deservește de asemenea și ca întâlnire de planificare: se vorbește despre ce s-a făcut de la ultima întâlnire și până acum și ce se va face de acum până la următoarea întâlnire; se discută și despre problemele și impedimentele care au afectat membri echipei în ceea ce privește atingerea obiectivelor stabilite. Conducătorul Scrum este cel care conduce această întâlnire ce durează aproximativ 15 minute. Fiecare întâlnire urmărește anumite reguli:
 - Întâlnirea începe la timp; de multe ori întârzierile se pedepsesc
 - Toți sunt bineveniți, însă doar cei implicați în procesul de dezvoltare au voie să vorbească
 - Întâlnirea durează 15 minute indiferent de numărul de participanți
 - Toți participanți ar trebui să stea jos
 - Toate întâlnirile ar trebui să aibă loc în fiecare zi în același loc și la aceeași oră
- **Întâlnirea de evaluare a unui sprint** – în ultima zi a sprint-ului Echipa împreună cu Conducătorul Scrum prezintă conducerii, clienților, utilizatorilor și Deținătorului de produs, rezultatul sprint-ului, într-o întâlnire neformală. Participanții la întâlnire evaluează rezultatul și iau decizii cu privire la direcțiile ce trebuie urmate în continuare. Durează cel mult 4 ore. Se pun două întrebări principale: “Ce a mers bine în timpul sprint-ului?” și „Ce se poate îmbunătăți în următorul sprint?”

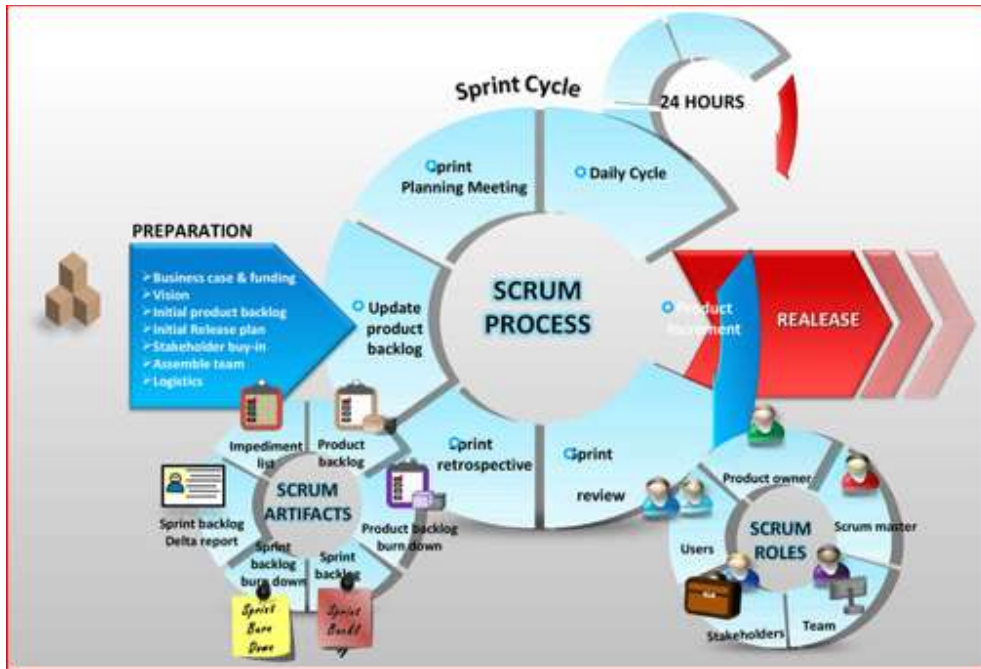


Figura 8.3.1. Procesul Sprint

Procesul de lucru include 3 faze: pre-joc, dezvoltare (sau joc), post-joc.

- **Pre-jocul** – include două sub-faze:
 - *Planificare*: presupune definirea sistemului ce se dorește a fi construit. Este creat un produs nerezolvat ce conține cerințele cunoscute la momentul actual. Cerințelor li se asociază priorități și este evaluat efortul necesar pentru implementarea acestora. Tot acum se stabilește și echipa ce va lucra la proiect, instrumentele și resursele necesare, ariile în care este nevoie de training.
 - *Arhitectura / Design la nivel înalt* – aici are loc design-ul de nivel înalt al sistemului; dacă sistemul deja există se discută schimbările necesare pentru implementarea cerințelor, precum și problemele pe care le ridică aceste schimbări.
- **Jocul** – reprezintă partea agilă a abordării Scrum. Această fază este tratată ca o cutie neagră unde unele elemente sunt imprevizibile. Variabilele identificate ce țin de mediul de dezvoltare, care se pot schimba de-a lungul timpului, sunt observate și controlate prin diverse practici Scrum. Scrum ia în considerare aceste variabile nu doar la început (în faza de pre-joc), ci pe tot parcursul procesului de dezvoltare pentru a se putea adapta ușor la schimbări. În această fază proiectul este realizat prin intermediul *sprint*-urilor.
- **Post-jocul** – este faza de finalizare a proiectului; toate cerințele stabilite au fost îndeplinite. În acest stadiu nu mai sunt elemente sau probleme (în produsul nerezolvat) ce trebuie adresate și nici nu se mai pot găsi altele noi. Produsul este gata pentru a fi livrat și se pregătește această acțiune (prin integrare, testare, documentare).

7. Metodologia de testare

7.1. Introducere

Fiecare proiect trebuie să conțină un plan amplu de testare care să cuprindă toate funcționalitățile aplicației și să asigure funcționarea corectă a întregului produs. Strategiile de testare se concentrează pe funcționalitatea și utilizabilitatea produsului.

Există câteva reguli care sunt considerate ca obiective ale testării:

- testarea este un proces de execuție a unui program cu intenția de a găsi o eroare;
- un caz de test bun este unul care are probabilitate mare de a găsi o eroare nedescoperită încă;
- un test terminat cu succes este un test în care se descoperă o eroare nedescoperită încă.

Metodele de verificare a corectitudinii unui produs se împart în două mari grupuri:

- metode statice de testare: constau în analiza unui program înainte de a fi lansat în execuție, independent de datele de intrare;
- metode dinamice de testare: constau în execuția programului; această metodă este cunoscută și sub numele de testare structurală.

Dintre cele mai familiare metode statice se amintesc: testarea specificației și examinarea codului.

La examinarea codului este inclusă și compilarea. Un compilator modern verifică tot felul de proprietăți ale programului, și refuză programele care nu respectă criteriile de corectitudine. Alteori compilatorul dă avertismente asupra unor construcții care generează probleme la execuție, cum ar fi de pildă variabile neinițializate.

Testarea sistemelor informatice complexe presupune testarea pe componente și părți ale componentelor (funcții sau clase), după un plan de test ce cuprinde toate cazurile de test. În sistemele complexe componentele sunt strâns legate între ele iar funcționalitatea unui modul este dependentă de alte module și atunci testarea se complică.

În cazul testării unui modul ce este dependent de informațiile din alte module se practică scrierea diferitelor date de test în baza de date pentru a testa toate cazurile posibile și rezultatul acestor teste se scrie în fișiere de log sau în baza de date.

Pentru un grup de module ce formează un subsistem se va descompune acest subsistem în funcții și module, pentru a se putea realiza testarea prezentă. Se testează mai întâi funcțiile apoi modulele și din aproape în aproape se agregă modulele și se obține testarea produsului finit.

7.2. Tehnici de testare software

7.2.1. Testarea arhitecturii Client/Server

Aplicațiile bazate pe arhitectura Client/Server sunt considerate aplicații complexe datorită numărului și diversității modulelor de prelucrare. Aceasta presupune o aplicație distribuită. În general o aplicație distribuită cuprinde un sistem central, mai multe subsisteme conectate la sistemul central și mai mulți clienți conectați la un subsistem. Complexitatea arhitecturii este reflectată și în testare.

Testarea acestei arhitecturi presupune trei niveluri diferite:

- client individual, caz în care aplicația client este testată individual, în mod deconectat și are ca efect acceptarea sau respingerea modulelor;

- client și server, caz în care acestea sunt testate împreună dar nu se ia în considerare rețeaua și are ca efect acceptarea sau respingerea interacțiunii client-server;
- client, server și rețeaua, caz în care se testează tot ansamblul împreună și se verifică dacă sistemul este corect, complet și funcționează în mediu real.

Sunt mai multe tipuri de teste care duc la detalierea acestor niveluri și anume:

- testarea funcționalităților;
- teste de server în care se verifică funcțiile de management al datelor și performanțele serverului;
- teste de baza de date în care se probează acuratețea și integritatea datelor salvate de server cât și metodele de arhivare;
- testarea tranzacțiilor în care se verifică clasele din punct de vedere al tranzacțiilor în conformitate cu cerințele; testul se focalizează pe corectitudinea procesului și pe performanțele acestuia.
- testarea comunicației în care se verifică traficul pe rețea, volumul datelor și corectitudinea lor; testele de securitate trebuie să facă parte din aceste teste.

7.2.2. Test de compatibilitate și configurare

Testarea compatibilității presupune verificarea interacțiunii software cu celelalte componente software cu care va coexista și va interacționa.

Testarea produsului pe mai multe platforme este o muncă foarte costisitoare, atât din punct de vedere al testării cât și din punct de vedere al rezolvării problemelor care sunt descoperite.

Testarea compatibilității, realizată în beta testing, reprezintă o testare externă cu un grup selectat ca potențiali clienți. Selectarea grupului se efectuează după criterii precise întrucât rezultatele testării sunt cu atât mai concludente cu cât există garanția că acești clienți selectați utilizează o diversitate cât mai mare de module.

O aplicație complexă are în componență multiple configurări de parametrii, de variabile de environment, setări dependente de sistemul de operare, de tipul bazei de date, de diferitele configurări pe care le permite aplicație în funcție de potențiali clienți. Foarte multe din aceste configurări se setează pe un singur calculator al clientului și se distribuie automat pe celelalte calculatoare.

Testarea configurării presupune verificarea îmbinării dintre setările posibile și ambientul software și hardware existent. Dacă aplicația este proiectată să folosească scanner trebuie să fie compatibilă cu hardware existent. Trebuie făcute setările necesare pentru a putea utiliza un anumit tip de hardware.

7.2.3. Testarea utilizabilității

Multe companii de software cheltuiesc foarte mult timp și bani pe găsirea celei mai bune căi de proiectare a interfeței cu utilizatorul, UI - *user interface*.

Pentru realizarea unei bune interfețe cu utilizatorul s-au desprins următoarele caracteristici importante:

- urmărirea utilizării unor standarde - De obicei standardele sunt dependente de sistemele de operare. Dacă o platformă nu are un standard sau se dorește adaptarea unui standard nou, atunci echipa de proiectare va crea un nou standard cât mai eficient;
- intuitiv - alegerea unui standard care să fie cât mai ușor de înțeles;
- consistența - folosirea consecventă a standardului ales în toate aplicațiile sau modulele dezvoltate: shortcut key, meniu, terminologie, numire, consistență în mesaje de erori, plasament al butoanelor fără a fi create confuzii, utilizând secvențe acumulate anterior;

- flexibilitate - să permită setarea diferitelor caracteristici ale aplicației dorite de clienți culori, valori default, diferite căi de introducere de date pentru a se potrivi cerințelor utilizatorilor și obiceiurilor lor;
- concret - să fie bine precizat și bine definit;
- utilitate - să fie folositor și să dea un randament bun, astfel duratele de prelucrare să fie reduse;
- accesibilitate - prietenos și ușor de utilizat prin mesaje comune cu alte produse și având grupe de taste de comandă cunoscute;

Ca parte a testării utilizabilității este și testarea documentației. Aceasta este de multe ori este considerată ca fiind o componentă non-software. Dar o documentație bună duce automat la creșterea utilizabilității produsului și implicit la creșterea operaționalității.

7.3. Strategii de testare

7.3.1. Unit testing

Termenul de 'testare unitara' se refera la testarea individuala a unor unitati separate dintr-un sistem software.

În timpul proiectării și codificării se comit erori care sunt grupate în următoarele categorii:

- erori legate de alegerea și descrierea algoritmului: algoritm incorect, sau corect dar inadecvat problemei; algoritm mai puțin performant ca precizie sau timp necesar rezolvării problemei; omiterea, interpretarea greșită sau incompletă a unor părți ale algoritmului; validarea incorectă și/sau incompletă a datelor de intrare; inversarea răspunsurilor la un bloc de decizie;
- erori în definirea și utilizarea datelor ce provin din variabile neinițializate, formate improprii de citire, contoare de capacitate insuficientă, neverificarea datelor de intrare, aliniere/redefinire incorectă a câmpurilor, utilizarea unor cuvinte cheie ca variabile, variabile ilegale (formate prin concatenare sau despărțite între două linii de program);
- erori de calcule care au ca surse: expresii complicate cu posibilități necontrolate de eroare; conversii implicite de tip (cu eroare de conversie, rotunjire, trunchiere); neinterceptarea cazurilor de depășire/subdepășire a intervalului definit;
- erori produse în tehnica de programare cum sunt variabile și structuri de date globale, acces necontrolat la zone de memorie partajate, interfețe program - subprogram nerespectate, pasarea constantelor ca parametri transmiși prin adresă, pasarea parametrilor de intrare/ieșire prin valoare, automodificarea programului în timpul execuției, utilizarea necontrolată a mai multor limbaje cu convenții de apel diferite;
- erori produse din neatenție caz în care logica de control e defectuoasă, salt în afara limitelor programului, condiții logice compuse sau incorect negate, neprelucrarea primei sau ultimei înregistrări, neluarea în considerare a posibilității de existență a fișierelor vide, neprelucrarea erorilor de intrare/ieșire, depășirea capacității stivei, adresare incorectă, necontrolarea indecșilor;
- erori în contextul execuției datorate memoriei dinamice insuficiente sau nealocată, periferice neoperaționale, comunicare defectuoasă cu sistemul de operare.

Cea mai mare parte a erorilor enumerate sunt depistate în faza de compilare a programului și sunt extrase în fișierul de ieșire, într-o formă specifică fiecărui compilator. Tot ca erori de codificare sunt considerate și cele detectate în faza de editare a legăturilor. În timpul execuției programelor apar erori de genul:

- erori de echipament, care sunt legate de contextul în care se execută un program și care se împart în: erori în datele de intrare, erori ce decurg din neglijarea specificului unui limbaj sau compilator (aritmetica numerelor în calculator, modul de implementare a tipurilor și structurilor de date pe un limbaj dat)
- erori de încărcare a programelor și de apelare incorectă a diferitelor periferice.

7.3.2. Testul de integrare

Are ca obiectiv testarea pe diverse niveluri de integrare a modulelor. Se pune accentul pe funcționarea corectă a ansamblului, pe compatibilitatea dintre componente, de asemenea se pune accent pe depistarea erorilor de interfață între module, păstrarea integrității semantice a structurilor de date externe, eliminarea conflictelor privind accesul la resursele de calcul.

Este indicat ca testele să fie desfășurate într-un mediu cât mai apropiat de cel în care sistemul va funcționa ulterior. Sunt practicate trei tipuri de modalități ale testării:

- testarea de sus în jos, top-down;
- testarea de jos în sus, bottom-up;
- testarea mixtă;

7.3.3. Testarea de sus în jos

Este folosită numai în cazul unei concepții descendente a sistemului. Metoda constă din următoarele: se pornește cu modulul rădăcină și cu unul sau mai multe niveluri de ordin imediat inferior; după testarea acestui schelet care probează toate posibilitățile legăturilor (interfețelor) se adaugă un alt nivel inferior; când s-au adăugat modulele ultimului nivel testarea este terminată.

Modulele de nivel superior apelează module fictive de nivel inferior, care vor fi implementate într-o anumită etapă, astfel:

- se prevede terminarea execuției lor dacă funcția pe care o realizează este corespunzătoare;
- ieșirile din aceste module se impun prin constante;
- se impun ieșiri aleatoare produse de generatoare de numere aleatoare;
- se tipărește un mesaj de avertizare pentru ca programatorul să fie informat că modulul vizat respectiv a intrat în execuție.

Se observă că testarea de sus în jos se desfășoară concomitent cu proiectarea și codificarea, adică: se proiectează programul principal, se programează și testează; apoi se proiectează, programează și testează programul principal împreună cu modulele de nivel imediat inferior, ș.a.m.d. până când ultimul nivel a fost proiectat, programat și testat. Această ultimă fază coincide cu testarea întregului sistem. Prin efectuarea testării în paralel cu proiectarea, pe total se reduce considerabil timpul de elaborare al unui produs program (uneori aproape cu o treime).

Deși principiul testării de sus în jos pare inaccesibil, există totuși o serie de avantaje: se elimină testarea întregului sistem, căci nu mai este necesară; se testează mai întâi interfețele dintre module, eliminându-se de la început erorile dificil de detectat; micile erori de codificare afectează numai un modul și deci se localizează cu ușurință; beneficiarii au o versiune preliminară parțial funcțională a programului; termenele de predare sunt respectate sau chiar devansate; timpul de testare este mai bine distribuit; testarea îmbunătățește moralul programatorului, întrucât oferă acestuia și beneficiarului rezultate parțiale pe parcurs.

7.3.4. Testarea de jos în sus

Este o metodă clasică de testare și constă în testarea individuală a modulelor urmată de testarea ansamblului de module ca un tot unitar.

Pentru fiecare modul trebuie efectuate: testul de funcționalitate se verifică dacă modulul îndeplinește funcția sau funcțiile sale, testul de depistare a datelor eronate ce nu corespund funcției și testul de comportare în condiții extreme de lucru.

Testarea sistemului nu este exhaustivă. Ea presupune verificarea sistemului din punct de vedere al specificațiilor sale, dar și al performanțelor: timp de răspuns, capacitate de lucru etc. Testele de sistem ocupă până la 30% din timpul total de realizare. Un dezavantaj al metodei este dificultatea în stabilirea datelor de test pentru sistemul final. Erorile cele mai dificile fiind cele legate de modul de comunicare între module date neutilizate, date inițializate în mai multe module, date modificate în diferite module, rutine suplimentare de simulare.

7.3.5. Metoda mixtă

Presupune aplicarea simultană a celor două metode precedente, rămânând predominantă testarea descendentă. Prin urmare se începe elaborarea proiectului într-o manieră descendentă, dar simultan se realizează module din nivelul de bază al ierarhiei. În acest caz testarea descendentă se desfășoară paralel cu cea ascendentă. Acest procedeu de concepere și testare s-a dovedit eficient în elaborarea multor produse program.

După cum se observă, nici una din metode nu trebuie impusă în mod rigid; fiecare suferă mici variații în cazul unor proiecte particulare. Sunt și situații când ordinea de abordare conține și zigzag.

7.3.6. Testul de validitate

Presupune o abordare graduală a modulelor specifice. Odată modulele asamblate și testate în integration test și după ce erorile de interfață au fost descoperite și rezolvate începe seria finală de test și anume *testul de validitate*. Validitatea este definită în diferite moduri dar o definiție simplă este că validarea este terminată când aplicația software funcționează într-o manieră rezonabilă acceptată de client. Acceptările rezonabile sunt definite în documentul ce cuprinde specificația cerințelor și care descrie toate atributele cerute de client.

Specificațiile conțin o secțiune numită criteriile de validare care reprezintă baza testului de validitate. Pentru realizarea acestui test se pregătește un plan de test împreună cu procedurile prin care se face testul. Planul și procedurile sunt proiectate astfel încât să se testeze cerințele, performanțele, documentația, compatibilitatea, întreținerea cât și procedurile de restaurare. În urma testului de validare se creează o documentație cu specificare pentru fiecare element din planul de test ca funcția/caracteristică de performanță:

- este conform cu specificația și este acceptată;
- nu este conform cu specificația și este atașată o listă de erori sau abateri de la specificație.

Pentru eliminarea erorilor descoperite în această etapă a proiectului se corectează înainte de predare dar de cele mai multe ori trebuie să se negocieze cu clientul pentru stabilirea metodelor pentru rezolvarea deficiențelor găsite în această etapă.

Un element important al procesului de validare este revederea configurației - configuration review. Se verifică dacă toate elementele necesare pentru configurare au fost dezvoltate și funcționează la parametri stabiliți. Este imposibil de imaginat cum clientul va folosi în realitate produsul realizat de o companie de software, chiar dacă acesta cuprinde un manual de utilizare.

Cei mai mulți producători de software utilizează procese numite Alpha Test și Beta Test pentru a descoperi erori pe care numai utilizatorii finali le descoperă.

Testul alpha se realizează de către clienți selectați, este condus de către dezvoltătorii de software și este de obicei într-un mediu controlat. Aplicația este utilizată având în spate dezvoltătorul pentru a înregistra erorile și problemele apărute.

Testul beta este făcut de unul sau mai mulți clienți finali fără nici un control din partea dezvoltătorului. Acesta este un test într-un mediu necontrolat (ambient real) în care clientul înregistrează toate problemele reale sau imaginare și vor fi raportate la intervale regulate către dezvoltător.

7.3.7. Testul de acceptare

Se efectuează cu scopul de a valida funcțional produsul din perspectiva utilizatorului final. Obiectivul recomandat este demonstrarea modului în care produsul se va integra în mediul de lucru real al beneficiarului. Ca alternativă, se urmărește familiarizarea utilizatorilor finali cu modul de operare a aplicației, caz în care are loc o trecere în revistă a funcțiilor aplicației. Se apreciază că acest tip de testare, care implică și participarea viitorilor beneficiari, este pentru dezvoltător o importantă sursă de informații privind contextul în care va fi utilizat sistemul.

7.3.8. Test de sistem

Este specific sistemelor complexe care trebuie să fie operaționale. Într-un sistem software complex este obligatoriu să se facă și testul de sistem.

Testul de sistem este compus dintr-o serie de teste al căror obiectiv este să testeze evoluția produsului software în condiții date de sistemul hardware. Avem următoarele teste care trebuie făcute în testarea sistemului:

- test de recuperare recovery testing;
- test de securitate security testing;
- test de stres stress testing;
- test de performanță performance testing.

Testul de recuperare

Este un test de sistem prin care se forțează sistemul să dea o varietate de erori pentru a putea verifica dacă restaurarea se realizează corect. Se verifică: restaurarea (automată sau manuală), reinițializarea, mecanismele de verificare a restaurării și respectiv timpul necesar pentru restaurare.

Testul de securitate

Presupune verificarea mecanismelor de protecție implementate în sistem, de fapt protecția la intrările neautorizate în sistem. Rolul unui proiect de securitate al unui sistem este să facă astfel încât costul de spargerea al sistemului să fie mai mare decât beneficiile pe care le obține prin spargerea sistemului.

Testul de stres

Presupune execuția sistemului într-o manieră anormală. Adică se testează confruntarea software cu situații anormale (multiple tranzacții, memorie insuficientă, spațiu liber mic pe disk, blocarea perifericelor cu care lucrează aplicația, etc.)

Testul de performanță

Este proiectat să testeze în run-time performanțele sistemului. Acest test se face atât la nivelul modulelor cât și la nivelul global al întregii aplicații, dar însă pentru verificarea cerințelor de performanță această testare se face după ce integrarea este completă. Testarea de performanță implică atât elemente software cât și elemente hardware.

7.3.9. Testare regresivă

Reprezintă o treaptă deosebit de importantă pentru echipele care doresc să dezvolte procese accelerate. Ca modalitate de lucru, prevede repetarea testării cu date de test și în condiții identice, pentru fiecare nouă versiune internă a unei componente software. Prin compararea rezultatelor testării și identificarea diferențelor se depistează erorile nou apărute; acest lucru este deosebit de util pentru maniera actuală de dezvoltare a aplicațiilor RAD - *Rapid Application Development*, care implică utilizarea instrumentelor vizuale de programare și se caracterizează prin apariția unui număr mare de modificări într-un interval scurt de timp.

Procesul de testare este asistat de instrumente specifice, care diminuează aspectele de rutină. Se apreciază că utilizarea instrumentelor de testare aduce beneficii comparativ cu efectuarea manuală a testelor, deoarece:

- testarea manuală, chiar în cazul unei planificări riguroase, prezintă riscul neidentificării erorilor din neatenție sau din cauza nerespectării riguroase a cazurilor de test prevăzute;
- testarea manuală solicită un consum intens de resurse umane, care sunt costisitoare și nu întotdeauna disponibile;
- testarea manuală este înceată comparativ cu testarea automatizată; adesea apare problema dezvoltării unor noi versiuni interne ale componentelor înainte de testarea completă a versiunilor precedente;

Dintre categoriile de instrumente pentru asistarea testării enumerăm:

- instrumente de capturare/redare înregistrează o sesiune de testare într-un fișier script, permițând repetarea acestuia și sunt efectuate teste multiple în manieră automată cu efectuarea de comparații asupra rezultatelor, aceste instrumente sunt eficiente în testarea regresivă;
- instrumente de execuție automată a testelor asemănătoare cu cele de mai sus, dar cazurile de test sunt specificate de utilizator în fișiere script;
- analizor de acoperire evaluează gradul în care structura codului testat a fost acoperită prin cazurile de test, astfel de instrumente sunt utile pentru identificarea porțiunilor de cod netestate;
- generator de cazuri de test este un instrument care, pe baza unor informații precum cerințe, modele ale datelor, modele obiectuale; generează cazuri de test semnificative, avantajul este eliminarea redundanței în testare, prin determinarea cazurilor de test care asigură acoperirea cât mai mare a codului; această activitate, executată manual, este dificilă;
- generator de date de test este un instrument care folosește la popularea fișierelor și bazelor de date în vederea testării, popularea se face în general cu date aleatoare, dar unele instrumente prevăd și posibilitatea specificării unor condiții; instrumentele sunt utilizate în general pentru popularea cu volume mari de date, necesare testărilor operaționale și la capacitate maximă;
- analizor logic / de complexitate servește la cuantificarea complexității unor porțiuni de cod; multe astfel de instrumente oferă și reprezentări grafice ale căilor posibile în structura codului; sunt utile pentru determinarea cazurilor de test necesare pentru atingerea anumitor puncte din cod din rutine complexe.
- instrumente de trasare a erorilor permit gestiunea informațiilor privitoare la erorile detectate și stadiul corectării lor și centralizarea acestor informații pentru urmărirea tendințelor acestor defecte; pe baza acestor tendințe se efectuează îmbunătățiri în procesele de dezvoltare și/sau mentenanță ale organizației;

- instrumente de gestionare a testării au rolul de a asista planificarea și organizarea elementelor implicate în testare precum fișiere script, cazuri de testare, rezultate;

7.3.10. Testare software orientat obiect

Obiectivul fundamental al testării rămâne neschimbat și în cazul software orientat obiect. Prin natura sa, programarea orientată obiect schimbă atât strategiile de testare cât și tacticile de testare. Se lucrează cu clase construcții incorporate polimorfe și ale căror proprietăți se moștenesc.

La nivelul programării orientate obiect testarea trece dincolo de identificarea de erori și atinge și laturi calitative ale definirii, referirii de clase și obiecte.

Prezintă importanță intensitatea cu care sunt referite clase deja existente în biblioteci. De asemenea testarea pune în evidență măsura în care sunt realizate nivelurile de încapsulare, moștenire și polimorfism. Strategiile de testare pentru software orientat obiect presupun o testare care începe prin testarea pe bucăți, părți componente după care urmează testarea în ansamblu. Deci se începe cu unit test se continua cu testul de integrare și se încheie cu testul de sistem și validare. Încapsulare conduce la definirea de clase și obiecte, instanțe ale claselor.

În programarea orientată obiect unit testing este echivalent cu testarea claselor - class testing. Fiecare clasă sau obiect împachetează date și metode cunoscute și ca operatori, care manipulează aceste date. Testarea claselor este o operație complexă datorită moștenirii claselor și redefinirii metodelor precum și tipurilor de moștenire privat, public sau protejat.

Programarea orientată obiect este caracterizată printr-un nivel foarte ridicat al reutilizării.

Testarea software orientată obiect presupune două planuri:

- testarea construcțiilor proprii;
- testarea construcțiilor incluse pentru reutilizare.

Pe lângă obiectivul general al stabilirii măsurii în care produsul software realizează sarcinile date în specificații, sunt și obiective speciale legate de:

- testarea funcțiilor membre ale fiecărei clase;
- testarea gradului de încapsulare și efectele acestuia;
- testarea efectelor induse de nivelul de moștenire și derivare;
- testarea efectelor induse de polimorfismul funcțiilor membre;
- testarea interacțiunilor dintre clase.

7.3.11. Metoda de testare empirică

Are un caracter parțial și se efectuează în etapele de analiză, proiectare, programare, integrare module. Este atât un proces de autoverificare cât și un proces global.

Testarea empirica se realizează în principal de către elaboratorii de programe și mai apoi de către utilizatorii programelor. Programul trebuie privit ca o cutie neagră. Din documentație, din module de proiectare a interfețelor rezultă structura datelor de intrare. Cum se efectuează prelucrările, care sunt acestea, ce efecte secundare sunt generate, nu reprezintă un element esențial din punct de vedere al testării empirice.

Obiectivul testării empirice este acela de a pune în evidență că programul e bun sau nu e bun (merge sau nu merge). Testarea empirică se focalizează în trei puncte și anume:

- la nivelul datelor de intrare, pentru a vedea dacă programul acceptă ca intrări datele care definesc problema; se testează situații cu date mai multe/mai puține și egale decât oferta.
- la nivelul prelucrărilor, în sensul traversării pașilor algoritmului execuției sau în puncte diferite, cu găsirea unor legături între datele oferite și punctul în care are loc întreruperea;
- la nivelul rezultatelor când se identifică rezultate incomplete structural, rezultate complete structural și incorecte și respectiv situația în care rezultatele corespund calitativ fără a putea fi făcute mențiuni asupra corectitudinii efective a lor.

Testarea empirică este direcționată fie spre latura pozitivă fie spre latura negativă a testării produsului. În urma testării empirice prin exemple de test se obțin rezultatele concrete prin care se definește comportamentul programului.

7.3.12. Metoda de testare ierarhică

Definește și aplică standarde de testare pentru câteva niveluri ale componentelor software: obiecte, clase, componente de bază și sisteme.

Metoda de testare ierarhică se axează pe componentele de bază. O componentă de bază este o ierarhie completă de clase sau anumite clustere de clase care realizează o funcție de bază sau care realizează o componentă arhitecturală logică sau fizică. Metoda de testare ierarhică desemnează ca fiind sigure acele componente care îndeplinesc standardele de testare pentru tipul respectiv de componentă. O componentă care a fost desemnată ca sigură este integrată cu alte componente sigure pentru a realiza împreună componentele de nivel următor.

Testarea de integrare a componentelor de bază sigure necesită accesarea doar a interconexiunilor dintre componentele de bază și orice funcționalitate complexă nouă.

7.4. Efectuarea testării

Testarea este unul din puncte cheie ale realizării unui produs software de calitate.

Testarea eficientă pentru un produs complex presupune și existența unor instrumente care asistă procesul de testare pentru a automatiza acest proces, ceea ce duce la creșterea costului testării.

Personalul necesar pentru testare trebuie să fie specializat să cunoască tehnicile de analiză, proiectare și programare și să înțeleagă problema pe care aplicația dorește să o rezolve. Procesul de testare se recomandă a fi independent de producător și de utilizator pentru a asigura rigurozitatea rezultatelor și a interpretării corecte a acestora.

Etapele testării se derulează astfel:

- se formează echipa de test în funcție de scopul testului și de aplicația de testat, cu cât sistemul software este mai complex cu atât crește numărul testărilor și specialiștilor;
- echipa va fi împărțită pe tipuri de funcții pe care trebuie să le testeze persoanele grupului de test;
- se construiesc exemplele de test și se utilizează și exemplele de test furnizate în specificație;
- se face un plan de test cuprinzând durata și numărul de iterații.
- se alege metoda de testare adecvată în raport cu produsul;
- se definesc documentele/rapoartele pe care trebuie să le elaboreze membri echipei de test, cât și documentele care se realizează la nivelul echipei;
- se colectează erorile, le stabilesc frecvența și se cuantifică efectele pe care acestea le generează la utilizatori;

- reproduc condițiile de producere a erorilor;
- în cadrul programării orientate obiect testarea trebuie să cuprindă în mod special testarea nivelurilor de încapsulare, moștenire și polimorfism, pentru fiecare existând tehnici de testare adecvate;

Cazurile de test trebuie să țină seama că:

- fiecare caz de test trebuie să fie identificat unic și asociat explicit cu clasele care vor fi testate;
- să se spună din start scopul testului;
- trebui realizată o lista de pași prin care trebui să treacă testarea ce trebuie să cuprindă:
 - listă cu stările prin care trebuie să treacă obiectul testat;
 - listă de mesaje și operații care trebuie făcute pentru ca testul să fie consistent;
 - listă de excepții prin care un obiect trebuie testat;
 - listă de condiții externe (exemplu: modificarea unor variabile de environment);
 - informații suplimentare necesare pentru a înțelege sau realiza testul.

Pe baza erorilor și documentelor colectare în urma testării, șeful de proiect va stabili timpii și prioritățile în rezolvarea defectelor.

7.5. Indicatori de testare

În cadrul etapei de testare se calculează diferiți indicatori care dau o imagine a procesului de testare cât și o imagine asupra calității sistemului testat.

Complexitatea testării reprezintă numărul de cazuri de test necesare raportat la volumul aplicației. Complexitatea testării se calculează atât pe întregul produs dar și pe o anumită parte sau unitate de produs. Mai exact complexitatea testării **C_t** reprezintă numărul cazurilor de test raportate la unitatea testată dată de relația:

$$C_t = \frac{CT}{UT}$$

unde: **CT** – Cazuri de test; **UT** – Unitatea testată;

Cazurile de test sunt raportate la nu anumit număr de tranzacții, un modul, un grup de module sau întregul sistem.

Calitatea defectelor raportate

Raportul dintre defectele efective și totalul defectelor raportate este numită calitatea defectelor raportate **C_d** dat de relația:

$$C_d = \frac{TD_{unice}}{TD} * 100$$

unde: **TD_{unice}** - totalul defecte unice; **TD** – totalul defectelor raportate.

Rata defectelor

Numărul de defecte efective descoperite relative la numărul cazurilor de test reprezintă rata defectelor **R_d**.

$$R_d = \frac{TD_{unice}}{CT} * 100$$

Rata incidentelor - numărul de incidente în exploatarea sistemului

Prin incident înțelegem acele erori care apar din exploatarea sistemului cum ar fi: afișarea și imprimarea rezultatelor, întreruperea programului, blocarea programului, resetări de funcții și de stări și alte comportamente neprevăzute. Acest indicator ne relevă eficiența măsurilor de întreținere și de corectare după implementare a produsului-program. Contorizarea acestor incidente se face pentru a calcula o rată a incidentelor R_i prin raportarea numărului de incidente la numărul de ore de funcționare sau la numărul de tranzacții efectuate date de relația:

$$R_i = \frac{N_I}{N_T} * 100$$

unde: N_I – număr de incidente; N_T – număr de tranzacții;

8. Intrarea în producție

În această etapă, produsul livrat va fi validat și acceptat de către beneficiar, iar utilizatorii au fost instruiți cu privire la utilizarea sistemului.

Orice defecte sau disfuncționalități identificate după acest moment fac obiectul contractului de garanție, iar dezvoltarea de funcționalități suplimentare presupune încheierea unor acorduri cu prestatorului.

Data intrării în producție a sistemului reprezintă data de la care curg termenele de garanție.

9. Instruirea utilizatorilor

Instruirea utilizatorilor sistemului se realizează diferențiat, în funcție de competențele fiecărei categorii de utilizatori, de preferință pe grupuri de utilizatori.

Procesul de instruire se realizează diferențiat, în funcție de nivelul de cunoaștere a fiecărui grup de utilizatori, conform metodologiilor de instruire standard în domeniu.

10. Asistență tehnică și suport post-implementare

În perioada de garanție și suport tehnic orice probleme vor fi remediate rapid, prin mijloace de asistență la distanță sau intervenții la sediul beneficiarului sau ale partenerilor acestuia.

Personalul tehnic și de implementare ce urmează a oferi activități de mentenanță și suport tehnic este instruit pentru a răspunde eficient și rapid la solicitările beneficiarului.

Dezvoltatorul pentru o perioadă de 12 luni după acceptarea sistemului informatic va oferi servicii de garanție și suport tehnic.

Serviciile de suport vor fi prestate de către Dezvoltator în vederea depășirii incidentelor produse în legătură cu utilizarea aplicațiilor de către BENEFICIAR și anume:

- Analiza defectului depistat/raportat cu scopul de a identifica eroarea;

- Introducerea corecturilor de rigoare în produsele software și efectuarea testării pentru confirmarea corectitudinii corectării erorilor raportate;
- Documentarea și raportarea corectărilor.

Operațiile serviciului de suport sunt oferite numai în timpul zilelor lucrătoare conform legislației Republicii Moldova, conform cerințelor 5/7, 8/24.

10.1 Clasificarea problemelor

Problemele sunt clasificate în patru nivele diferite de Gravitate în dependență de impactul pe care îl au asupra Aplicațiilor și funcțiilor Aplicațiilor.

Gravitate 1 - Critică

Problema va fi definită ca nivel de gravitate 1 când ea produce o situație de avarie în care Aplicația este imposibil de utilizat, și nu există metodă de ocolire pentru Beneficiar (defectarea completă de întreruperile în funcțiile principale ale Aplicație software respective). Când au loc întreruperi extrem de serioase în operațiile normale și sarcinile nu pot fi executate, Dezvoltatorul va lucra asupra soluției tehnice până ce nu va fi implementată o Soluției de ocolire.

Această categorie include orice defect considerabil (sau altă nefuncționare considerabilă a Software conform Specificațiilor sale) care poate fi demonstrat și cauzează inoperabilitatea Software, operarea necorespunzătoare sau produce rezultate considerabil diferite de cele descrise în documentația care previn sau serios diferă de funcționarea aplicațiilor sau funcțiilor majore ale aplicațiilor. Un număr semnificativ de utilizatori ai aplicațiilor la moment nu-și pot îndeplini sarcinile necesare.

Gravitate 2 - Înaltă

Problema va fi definită ca nivel de gravitate 2 când: (i) componente semnificative, dar nu primare, ale Aplicației este imposibil de utilizat sau nu funcționează conform specificațiilor aplicabile; sau (ii) problema cu nivelul de gravitate 1 nu este capabilă a fi reprodusă 100%, dar are loc deseori. În acest caz, Dezvoltatorul va veni cu resursele necesare tehnice pentru a oferi soluționarea erorii sau metodei de ocolire.

Problema cu gravitatea 2 este cauzată când au loc întreruperi serioase în operarea obișnuită. Nu pot fi efectuate sarcini importante. Aceasta este cauzat de defectarea sau funcționarea indisponibilă în Aplicației software respective care necesită prelucrarea situației curente.

Gravitate 2 înseamnă o problemă majoră în care Aplicația suferă problemă din cauza pierderii funcționării și/sau funcționalității. Problema care semnificativ afectează abilitatea BENEFCIARULUI de desfășurare a afacerii/activității operaționale, gravitatea căreia este semnificativă și se poate repeta și are un impact asupra funcționării oportune a funcțiilor operaționale. De asemenea este influențată funcționarea Aplicației.

Gravitate 3 - Medie

Problema va fi definită ca gravitate 3 când ea produce situație inconvenabilă în care Aplicația poate fi utilizată, dar nu oferă funcționarea în cel mai corect mod. În acest caz, Dezvoltatorul va veni cu resursele necesare pentru a oferi soluționarea erorii sau metodei de ocolire.

Problema cu gravitatea 3 este cauzată când au loc întreruperi în operarea normală. Aceasta este cauzată de defectarea sau indisponibilitatea funcției în Aplicația software respectivă.

Gravitatea 3 înseamnă o problemă în care Software suferă o problemă ce cauzează pierdere de funcționare și / sau funcționalitate. Problema care influențează minim abilitatea BENEFICIARULUI de desfășurare a afacerii / activității operaționale , dar poate funcționa prin ocolire.

Gravitate 4 - Scăzută

Problema va fi definită ca gravitate 4 când au loc numai întreruperi minore în operările normale. Aceasta este cauzată de defectarea sau indisponibilitatea funcției în Aplicație software respectiva care nu este necesara zilnic sau nu este folosita regulat.

Gravitate 4 înseamnă o cerere de îmbunătățire generală. Impactul asupra inconvenienței BENEFICIARUL și impacte nignlijabile asupra abilității BENEFICIARUL de desfășurare a afacerii/ activității sale operaționale.

10.2. Soluționarea problemelor la distanță

Inginerul Dezvoltatorului, conform aprobării preliminare a BENEFICIARULUI referitor la asigurarea din partea BENEFICIARULUI a accesului la distanță, va efectua diagnosticare la distanță, când asistența la telefon și descrierea și registrele oferite în ticket sunt insuficiente sau total nereușite pentru identificarea și izolarea problemei în privința unei părți concrete a Aplicațiilor.

Lipsa de acces la distanță, din motive atribuibile BENEFICIARULUI, submină abilitatea Dezvoltatorului de garantare a soluționării conform valorilor de timp stabilite mai jos. Timpul scurs între cererea pentru acces la distanță și oferirea efectivă a accesului la distanță va fi scăzut din timpul stabilit în tabelul de mai jos.

10.3. Serviciul timpurilor tratării problemelor

Pentru serviciul de suport și anume: suport tehnic funcțional, managementul accesului și restaurarea serviciului, Dezvoltatorul va trata problemele conform termenilor de timp de mai jos:

Tabelul de mai jos definește limitele de timp a problemelor ce vor fi aplicate și respectate pentru livrabilele dezvoltate:

-	Timpul răspunsului	Timpul soluționării
Severitate 1	până la 2 ore	până la 5 ore
Severitate 2	3 ore	până la 8 ore
Severitate 3	8 ore	până la 5 zile
Severitate 4	12 ore	până la 5 zile

Toate valorile de timp vor fi în conformitate cu cele înregistrate pe web portal (Das Soft - eService) scăzând toate valorile de timp când progresul se află în controlul BENEFICIARULUI.

10.4. Orarul serviciilor

Pentru suport tehnic funcțional, managementul accesului și restaurarea serviciului a modulelor inginerii Dezvoltatorului vor fi disponibili:

De luni până vineri de la orele 9:00 până la orele 17:00

10.5. Raport de activitate pentru servicii de mentenanță și suport

- Dezvoltatorul va prezenta raport de activitate lunar pentru problemele raportate în decursul perioadei, care va și include progresul de soluționare pentru fiecare problemă
- Dezvoltatorul va prezenta un plan săptămânal de activitate pentru problemele planificate a fi soluționate ținând cont de prioritatea fiecărei probleme raportate.

11. Metodologie de analiză și combatere a riscurilor

Managementul riscurilor este o componentă majoră a managementului de proiecte. Printr-o abordare pro-activă, asigurăm rezultate optime prin reducerea probabilității de materializare și diminuarea impactului riscurilor și vulnerabilităților ce pot afecta bunul mers al proiectelor implementate.

Managementul riscurilor este o activitate curentă. Ședințele de lucru și ședințele tehnice au o secțiune dedicată analizei de risc, identificării riscurilor potențiale sau materializate, elaborării strategiilor de contracarare și diminuării vulnerabilităților.

Riscurile sunt identificate astfel, în etape incipiente ale proiectului. Identificarea surselor de risc se realizează începând cu etapa de analiză a specificațiilor proiectului (în special prin brainstorming) și reprezintă punctul de pornire al întocmirii registrului riscurilor. Cu toate acestea, identificarea prealabilă a tuturor riscurilor este imposibil de realizat. Astfel, este necesar ca managerul de proiect în primul rând, managerul de risc și toți membrii echipei de analiză, dezvoltare și implementare să sesizeze rapid potențialele riscuri și să contribuie la identificarea mijloacelor de contracarare.

O componentă de o importanță deosebită în managementul riscurilor îl este comunicarea. Membrii echipei avertizează managerul de proiect sau managerul de risc, care, la rândul său, ia măsuri în vederea diminuării probabilității de materializare și, dacă este cazul comunică cu beneficiarul în vederea stabilirii unei strategii comune de acțiune.

De asemenea, menținerea unui registru al riscurilor deschis reprezintă un instrument util atât pentru echipa internă cât și pentru beneficiarul final.

Managementul riscurilor are adesea conotații negative. Cu toate acestea, uneori materializarea unor riscuri poate genera oportunități noi pentru proiectul în curs de desfășurare. Astfel, printr-un management eficient al schimbărilor (riscuri pentru implementarea planului inițial de proiect), rezultatele pot adesea fi atinse mai rapid și mai ușor.

Riscurile sunt clasificate în funcție de deținătorul riscurilor. Astfel, unele riscuri pot fi gestionate doar de anumite persoane/entități, iar alte riscuri pot fi gestionate de oricare membru al echipei de implementare. Managementul riscurilor specifice beneficiarului cade în sarcina acestuia. Managerul de proiect sau managerul de risc din cadrul echipei noastre sprijină beneficiarul în gestiunea acestor riscuri.

Fiecărui risc îi este asociată o prioritate. În funcție de probabilitatea de producere și impactul în cazul producerii, tratarea unor riscuri este considerată prioritară sau nu.

Analiza riscurilor este o activitate permanentă, care are rolul de a ne ajuta să înțelegem natura riscului în vederea identificării celor mai adecvate mijloace de răspuns. Pentru fiecare risc sunt identificate componentele afectate și efectele asupra proiectului în ansamblul său.

După identificarea, priorizarea și analiza riscurilor, este elaborat planul de răspuns la riscuri. Acesta conține mijloacele și acțiunile ce trebuie întreprinse în vederea diminuării efectelor unui risc, în situația în care acesta se materializează.

Sunt utilizate trei abordări în ceea ce privește răspunsul la riscuri: prevenirea, diminuarea efectelor și acceptarea. Prevenirea presupune asumarea de contramăsuri astfel încât materializarea riscului să devină improbabilă. Diminuarea riscului presupune reducerea probabilității de materializare a riscului până la un punct, precum și măsuri de diminuare a efectelor în situația în care acesta se produce, prin intermediul unor planuri de rezervă sau contingență. Acceptarea riscului este răspunsul preferat atunci când probabilitatea de producere a riscului este relativ redusă, iar costurile sau resursele consumate pentru prevenirea materializării sunt în general mai mari decât pagubele provocate în situația în care riscul se materializează.

Pentru fiecare proiect, și pentru sub-proiectele importante constituim Registrul Riscurilor. Acesta este un instrument util echipei de implementare și adesea beneficiarului. Registrul riscurilor are următoarea structură:

- Numărul riscului
- Tipul riscului
- Data identificării
- Data ultimei revizuirii
- Descrierea
- Probabilitatea de apariție
- Severitatea
- Impactul
- Recomandări și strategii de diminuare/eliminare
- Responsabilul cu verificarea implementării contra-măsurilor
- Starea riscului (deschis, închis)

Managementul riscurilor este o activitate continuă în toate etapele de dezvoltare a proiectelor. Mai mult, etapa de dezvoltare bazată pe metodologii agile contribuie pozitiv la reducerea riscurilor specifice, iar ședințele periodice (cu periodicitate ridicată) aduc beneficii semnificative în ceea ce privește identificarea rapidă a riscurilor.

În cazul acestui proiect, Managerul de Proiect are sarcina de a gestiona managementul, fiind totodată și un Manager de Riscuri.

Managementul riscurilor se realizează prin patru activități:

- **Planificare** - constă în identificarea resurselor necesare derulării acțiunilor stabilite în faza de analiză, în dezvoltarea unui plan de acțiune și includerea acestui plan în cadrul Planului de Etapă și în obținerea aprobării pentru acest plan;
- **Alocare corespunzătoare a resurselor** - constă în identificarea și alocarea resurselor care vor fi utilizate pentru a acționa în scopul evitării riscului sau a minimizării impactului sau; aceste alocări vor fi incluse în Planul de Etapă; resursele necesare pentru întreprinderea acțiunilor de prevenire, reducere și transfer vor fi suportate din bugetul proiectului;

- **Monitorizare** - constă în verificarea faptului că acțiunile planificate și puse în practica au efectul dorit asupra riscurilor identificate; examinarea semnalelor timpurii de apariție a riscului; prognozarea riscurilor potentiate; verificarea faptului că **Managementul riscului** se realizează într-un mod eficient;
- **Control** - adică acțiunile care se iau și prin care se verifică faptul că planurile sunt respectate.

Metodologie definire Riscuri

Evenimentele (amenințările) care ar putea afecta proiectul au fost identificate. Riscurile aduse de aceste amenințări proiectului au fost cuantificate prin probabilitate și impactul adus folosind următoarea metodologie:

IMPACT - Impactul negativ pe care evenimentul l-ar putea avea asupra proiectului.

NIVEL	SCOR	DESCRIERE
SCĂZUT	1-5	Nu are un impact semnificativ asupra proiectului.
MEDIU	6-14	Impact semnificativ asupra proiectului. Cu toate că unele aspecte ale proiectului (de ex. borne interne) ar putea fi afectate, obiectivele principale sunt totuși îndeplinite.
RIDICAT	15-20	Termenul final sau obiectivele proiectului nu sunt îndeplinite.

PROBABILITATE – Probabilitatea ca o anumită amenințare să fie întâlnită pe durata proiectului.

NIVEL	SCOR	DESCRIERE
FOARTE SCĂZUT	1	Probabilitate foarte scăzută de apariție
SCĂZUT	2	Evenimentul s-ar putea întâmpla dar este puțin probabil
MEDIU	3	Evenimentul s-ar putea întâmpla pe durata proiectului
RIDICAT	4	Evenimentul e probabil să se întâmple
FOARTE RIDICAT	5	Evenimentul este aproape sigur să apară

Nivelul de risc este cuantificat după cum urmează:

NIVEL RISC = IMPACT x VALOAREA PROBABILITĂȚII

Nivelul de risc este cuantificat luând în considerare că nu au fost implementate măsuri de contracarare.

NIVEL DE RISC	VALOARE	MĂSURI NECESARE
---------------	---------	-----------------

MAXIM	75 – 100	Acțiune imediată pentru reducerea nivelului de risc
GRAV	25 – 74	Implementarea de măsuri corective/preventive cât mai repede posibil
MEDIU	5 - 24	Implementarea de măsuri corective/preventive într-o perioadă de timp
SCĂZUT	2 - 4	Se pot implementa măsuri corective sau se accepta riscul
FOARTE SCĂZUT	1	Se acceptă riscul

După ce sunt aplicate măsurile de contracarare, se obține riscul rezidual. Vom accepta doar acele riscuri care au cel mult un nivel rezidual SCĂZUT.

Riscurile proiectului și planul de tratare al acestora

Va rugăm să găsiți în tabelul de mai jos principalele riscuri identificate în desfășurarea proiectului și planul de diminuare și tratare al acestora:

Nr.	Risc inițial	Nivel risc inițial	Tratament	Contramăsuri necesare	Nivel risc rezidual
1	Sistemul software dezvoltat nu va îndeplini așteptările Beneficiarului	Mediu	Diminuare	Se va întocmi Analiza de Business detaliată care va îngloba toate așteptările Beneficiarului, document ce va fi semnat de către ambele parti. Se vor face ședințe de progress regulat cu Managerul de Proiect din partea Beneficiarului	Scăzut
2	Deadline nu este respectat	Mediu	Diminuare	Se va urmări fiecare deadline intermediar pentru a se putea face corecții până la finalul proiectului. Se va comunica permanent cu toate părțile implicate în proiect iar când una dintre părți nu va putea livra (de exemplu deținătorul unui sistem terț) acesta va	Scăzut

Nr.	Risc inițial	Nivel risc inițial	Tratament	Contramăsuri necesare	Nivel risc rezidual
				înștiința oficial asupra motivelor sale. Se va urmări planul de proiect și orice solicitare va avea un termen de răspuns agreat (cel mai probabil nu mai mult de 1-2 zile).	
3	Termenii intermediari de livrare pe etape nu vor fi respectați	Mediu	Reducere	Orice întârziere a unei etape va diminua din timpul alocat etapei următoare. Se va comunica constant și se va încerca recuperarea întârzierilor în caz ca vor exista.	Foarte Scăzut
4	Personalul cheie al proiectului din partea companiei va părăsi echipa.	Mediu	Diminuare	Se va asigura backup al tuturor persoanelor implicate în proiect având competențe similare.	Foarte Scăzut
5	Schimbări în sistemul legislativ sau în climatul politic și economic care ar putea afecta proiectul.	Foarte Scăzut	Acceptat	Dat fiind perioada foarte scurta de implementare, riscul este acceptat	Foarte Scăzut
6	Beneficiarul va schimba cerințele după începerea etapei de dezvoltare a sistemului software.	Scăzut	Reduce	Acest lucru nu se va putea întâmpla deoarece Beneficiarul va accepta Analiza de Business în forma ei finală. Orice neclaritatea din partea Beneficiarului va fi comunicată în această etapă.	Foarte Scăzut
7	Sistemele software terțe care trebuie integrate au bug-uri necunoscute la acest moment.	Ridicat	Diminuare	Se vor efectua 3 tipuri de teste: - Testarea Dezvoltatorului - Testarea Coordonatorului Tehnic - Testarea de acceptanță a Beneficiarului	Mediu

Nr.	Risc inițial	Nivel risc inițial	Tratament	Contramasuri necesare	Nivel risc rezidual
				Orice defecțiune a sistemelor terțe vor fi comunicate de Prestator în timp util iar termenul de remediere din partea detinatorilor acestor sisteme va fi agreat prin contractul dintre Beneficiar și Prestator.	
8	Softul dezvoltat va avea vulnerabilități de securitate	Mediu	Diminuare	Prestatorul va rula teste interne de securitate și le va remedia pe parcursul perioadei de testare	Scăzut

Soluții de preîntâmpinare a riscurilor

Este nevoie de a evita posibilitatea de a înțelege greșit nevoile clientului. Reducerea acestui risc se poate face prin:

- utilizarea unor modalități precum citirea unor rapoarte scrise asupra unor probleme, pentru a identifica nevoile clientului;
- prezența a cel puțin doi membri ai echipei la fiecare întâlnire cu personalul beneficiarului;
- discuții cu diferite persoane din conducerea beneficiarului;
- notarea conținutului comunicărilor;
- împărtășirea progreselor evaluate cu beneficiarul, la fiecare două săptămâni, pe durata proiectului

12. Echipa de proiect

Pentru a va oferi serviciile specificate, am constituit o echipă cu o vastă experiență în acest domeniu. Vă prezentăm în continuare membrii cheie ai echipe din cadrul companiei Das Soft Plus S.R.L care va fi implicată în acest angajament.

Echipa de bază este formată din 5 membri:

- ❖ 1 Project Manager
- ❖ 2 Dezvoltatori de Software
- ❖ 1 Inginer de Asigurare a Calității Software

Nume: Denis Dumitras

Poziția: Project Manager

Experiență: 5 ani

Puncte forte:

- Înțelegerea profundă a tehnologiilor software și a tendințelor pieței în domeniu
- Abilitatea de a comunica și a negocia eficient cu membrii echipei, cu clienții și cu alte departamente ale companiei.
- Capacitatea de a planifica și de a gestiona proiecte complexe de dezvoltare software, respectând termenele și bugetul alocat.
- Capacitatea de a analiza datele și informațiile complexe, pentru a face decizii înțelepte și informate privind produsul.
- Abilitatea de a dezvolta o viziune strategică pentru produs și de a implementa acțiuni pentru atingerea obiectivelor.

Nume: Mihai Dascăl

Poziția: Arhitect de Sistem / Dezvoltator de Software

Experiență: 13 ani

Puncte forte:

- Capacitatea de a transforma cerințele de afaceri în soluții tehnice
- Capacitate dovedită de a livra rezultate și de a stimula creșterea afacerii
- Experiență distinctivă în dezvoltarea aplicațiilor de sănătate (Web, Mobile, APIs), arhitectură de proiect, management de proiect și DevOps.
- Cunoștințe solide despre tehnologiile inovatoare de dezvoltare software.

Nume: Alexandru Negruta

Poziția: Dezvoltator de Software

Experiență: 6 ani

Puncte forte:

- Experiență remarcabilă în dezvoltarea aplicațiilor de sănătate
- Abilitate demonstrată în tehnologiile inovatoare de dezvoltare software
- Cunoștințe solide despre proiectarea, dezvoltarea și managementul bazelor de date
- Abilități avansate de depanare și remediere a problemelor
- Capacitatea dovedită de a lucra într-un mediu orientat spre echipă.

Nume: Sergiu Rusu

Poziția: Inginer de Asigurare a Calității Software

Experiență: 7 ani

Puncte forte:

- Cunoștințe solide despre metodologiile și practicile de testare a software-ului
- Experiență excepțională în testarea automatizată utilizând diferite instrumente și framework-uri
- Capacitatea de a crea și implementa planuri și strategii de testare
- Abilități dovedite de a dezvolta și menține scripturi de testare
- Abilități avansate de depanare și remediere a problemelor

ANEXE

- **CV-urile echipei de proiect**