Raport privind integrarea testelor de framework privind vizualizare și monitorizare

Raport științific și tehnic 2021

28.05.2021

Istoricul versiunilor

Istoricul versiunilor			
Autor	Commentarii	Versiune	Data
Oliviu Matei	Schiță	0.1	
Daniela Delinschi, Rudolf Erdei	Schiță	0.3	
Oliviu Matei, Daniela Delinschi, Rudolf Erdei	Final	1.0	2021.05.31

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Cuprins

1.Introducere 6
2. Arhitectura software 7
3. Abordare 9
3.1 Cartografierea procesului de dezvoltare 9
3.2 Strategia de testare 9
3.2.1 Caracteristici de calitate pentru fiecare nivel de test 10
4. Testarea principiilor de implementare 12
4.1 Trasabilitate 12
4.2 Activități de testare 12
4.2.1 Completare 13
4.2.2 Criterii de intrare 13
4.2.3 Criteriul de acceptare 14
4.2.4 Criterii de validare 15
4.3 Testarea regresiei 15
4.3.1 Regresia manuală 15
4.3.2 Regresie automată 15
4.4 Raportarea problemelor 16
4.4.1 Instrucțiuni 16
4.4.2 Șablon de probleme 16
5. Testarea unitară 17
5.1 Scop 17
5.2 Domeniul de aplicare 17
5.3 Abordare 17
6. Testarea integrării 17
6.1 Scop 17
6.2 Domeniul de aplicare 18
6.3 Abordare 18
7. Testarea sistemului 20
7.1 Scop 20
7.2 Domeniul de aplicare 20
7.3 Abordare 20
8. Testare nefuncțională 21
8.1 Scop 21
8.2 Domeniul de aplicare 21

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

8.3 Fiabilitate și securitate 22
8.3.1 Acoperirea codului 22
8.3.2 Interpretare abstractă 22
8.3.3 Avertismente ale compilatorului 22
8.4 Testabilitate 22
8.4.1 Complexitatea ciclomatică 22
8.4.2 Modularitate 23
8.5 Mentenabilitate 23
8.5.1 Standarde de codare 23
8.5.2 Duplicarea codului 23
8.5.3 Cod mort 23
8.6 Relații metrice 24
9. Integrare continuă 25
9.1 Scop 25
9.2 Practica obișnuită 25
10. Metodologia de validare 26
10.1 Definirea cerințelor 26
10.2. Validarea cerințelor 27
10.3. Rafinarea cerințelor BC 27
10.4. Definirea testelor de acceptare a utilizatorului (UAT) 27
10.5. Validarea arhitecturii în funcție de cerințe 27
10.6. Validarea modelul pe baza UAT 28
11. Metodologia de testare 28
11.1. Analiza codului 28
11.2 Setul de instrumente pentru fiabilitate 31
11.2.1 Securitate 31
11.2.2 Punct de control optim 34
11.2.3 Analiza rezultatelor 35
11.3 Testare funcțională vs. nefuncțională 36
12. Concluzii Raport tehnic 44
13. Artefacte 45
14. Livrabile 46
15. Articole publicate 46
16. Alte activități de diseminare 47
Referințe 49

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

1.Introducere

Transportul multimodal este definit de Convenția ONU privind transportul internațional multimodal de mărfuri după cum urmează. Transportul multimodal înseamnă transportul mărfurilor dintr-un loc într-un alt loc, de obicei situat într-o altă țară, prin cel puțin două mijloace de transport (Z. Peplowska-Dabrowska, 2019).

De obicei, când ne gândim la transport, ne gândim la o legătură directă (Figura 1 stânga), o cale mai scurtă, între emițător și receptor (K. G. Zografos, 2008).

Optimizarea acestor rute este un subiect destul de vast și intens studiat. Există deja algoritmi dedicați, precum Algoritmul lui Dijkstra (Y. Y. G. Jianya, 1999) pentru generarea unei singure rute sau tehnica Clarke-Wright (B. L. Golden, 1977) pentru abordările VRP. Dar aceste abordări au un lucru în comun: țin seama doar de un singur mijloc de transport, cu un singur depozit și unul sau mai mulți clienți (sau destinatari).

În familia sistemelor de transport inteligente (ITS), sistemele de transport multimodal (MMTS) s-au plasat ca mijloc principal de transport în timpurile noastre. Economia globală a progresat, doar cu ajutorul transportului pe scară largă. Volumul mărfurilor și distanțele parcurse s-au dublat în ultimii 10 ani, deci există o cerere mare pentru a face transportul global mai rapid, mai ieftin, mai sigur și cel mai important, mai ecologic.

Contextul actual al globalizării mondiale a ridicat multe probleme dificile în ceea ce privește transportul de mărfuri. Aceste mărfuri sunt transportate pe distanțe foarte mari de pământ și apă, coletele foarte mici ajung să meargă la mii de kilometri până la destinație și mai des ajung să călătorească cu mai multe mijloace de transport: cu nave, avioane, camioane. (W.-J. Van Schijndel, 2000)

Acest lucru duce la nașterea sistemelor de transport multimodal (MMTS). Spre deosebire de transportul clasic, unic, transportul multimodal are constrângeri multiple (T. Litman, 2017), de exemplu, diferite procese de optimizare, cum ar fi încărcarea coletelor și transferul între transporturi.

(Steadie Seifi, et al, 2014) în cadrul revizuirii literaturii de specialitate privind planificarea transportului multimodal de mărfuri, arată mai multe probleme de planificare strategică în cadrul transportului de mărfuri multimodal și probleme de planificare tactică. Sunt descrise planificări operaționale complexe pentru cerințele în timp real ale operatorilor, transportatorilor și expeditorilor multimodali, care nu au fost abordate anterior la niveluri strategice și tactice. Sunt incluse principalele modele cu soluțiile conexe și cercetările viitoare propuse.

Zamanifar și Hartmann (M. Zamanifar, 2020) furnizează o analiză detaliată a modelelor de luare a deciziilor bazate pe optimizare pentru problema Planificării recuperării dezastrelor a rețelelor de transport (DRPTN).

Cu toate acestea, în lumea reală, mărfurile pot fi transportate în orice direcție, de exemplu în interiorul unei țări există curieri care livrează în 24 de ore din orice punct al țării către orice alt punct (Figura 1 dreapta). Folosirea doar a camioanelor pentru a efectua, de exemplu, o sarcină completă ar fi aproape imposibilă, deoarece problema are o complexitate $O(n^2)$.

De exemplu, într-o țară mică, cu doar 20 de orașe, în fiecare zi ar exista 380 de camioane în mișcare. Optimizarea acestui caz ar însemna proiectarea unui sistem cu un depozit central (sau hub), unde toate mărfurile sunt descărcate, sortate în funcție de destinație și încărcate în cele din urmă pe camioanele respective și expediate către destinație.

Numai că această optimizare reduce numărul necesar de camioane de la 380 la doar 20 pentru aceeași problemă (același camion face o călătorie dus-întors din fiecare oraș către

hub-ul central) (M. Zhang, 2013). Dar cum rămâne cu țările mai mari sau cu transportul internațional?

A avea un singur hub pe un întreg continent ar putea să nu fie o idee prea bună, din cauza numărului foarte mare de rute de transport posibile. De asemenea, limitându-ne doar la camioane ar însemna că clienții de pe insule sau peste mări nu pot fi deserviți. Deci, livrarea produselor cu multe mijloace de transport, și anume transportul multimodal este o necesitate. Acest mod are avantajul de a muta o cantitate uriașă de mărfuri, în sute de mii de tone simultan, prin nave mari, pe distanțe foarte mari.

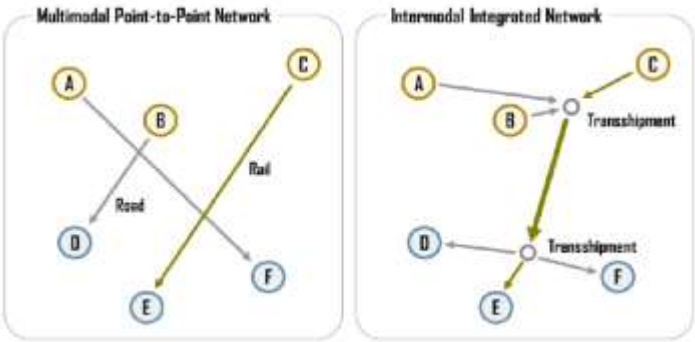


Figura 1 Diferența dintre o rețea multimodală

2. Arhitectura software

Datele pot proveni de la:

- modelul ontologic (a se vedea livrabilul *D2.2. Ontologia logistică*);
- datele proprii colectate de-a lungul timpului;
- surse de date de la terță parte, cum ar fi parteneri, date publice sau de la senzori.

Arhitectura software la nivel înalt este prezentată în

Figura 2:

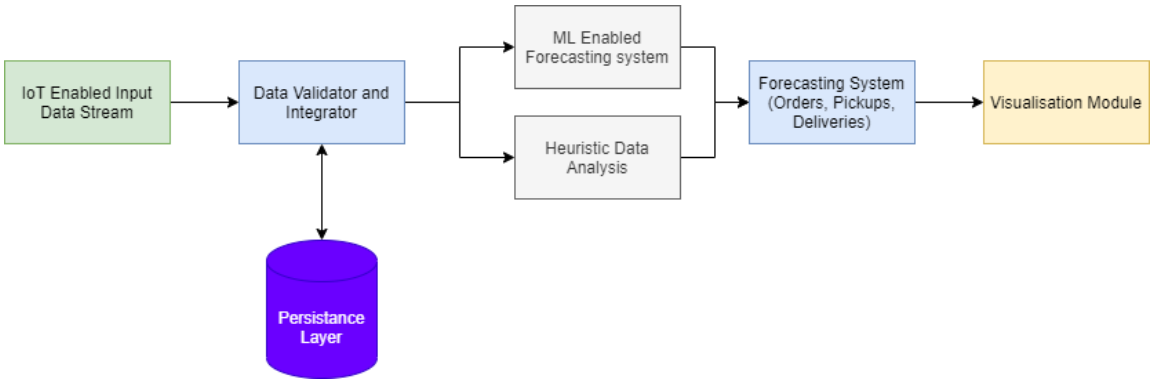


Figura 2 Arhitectura software de nivel înalt pentru Big-Smart-Log

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

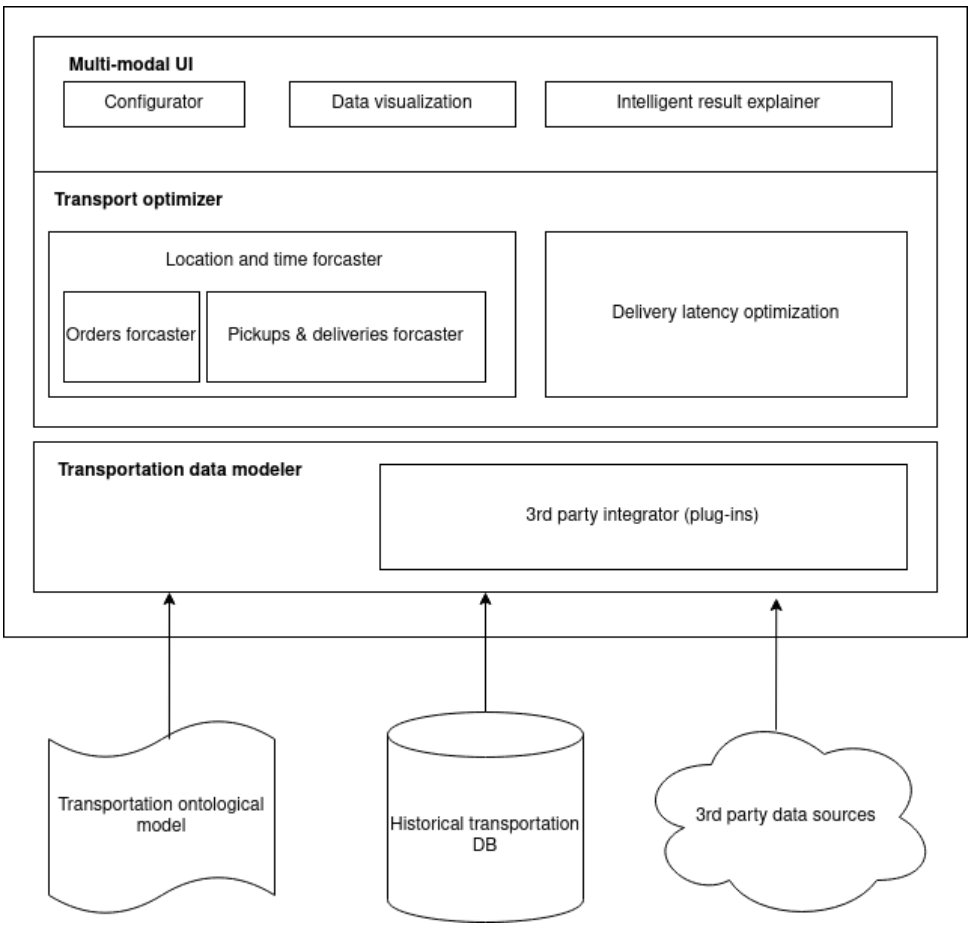


Figura 3 Arhitectura software detaliată pentru platformă

Datele sunt agregate de modelatorul de date transport, care le convertește într-un format standard, care este utilizat în continuare pentru analiză și vizualizare. Acest strat - Modelatorul de date de transport, separă stratul de date de stratul de business intelligence. Datele de la părțile terțe pot fi colectate folosind modele de plug-in (Richards, 2015). Procesul de transformare a datelor este cunoscut sub numele de extract-transform-load (ETL) (Vassiliadis, 2009).

Extragerea datelor presupune extragerea datelor din surse omogene sau eterogene; transformarea datelor; procesarea datelor prin curățarea datelor și transformarea acestora într-un format / structură de stocare adecvată în scopul interogării și analizei; în cele din urmă, încărcarea datelor descrie inserarea datelor în baza de date țintă finală, cum ar fi un depozit de date operațional, un data mart, un lac de date (data lake) sau un depozit de date (Henry, 2005). Procesul poate avea uneori sublinieri ontologice (Bansal, 2014), ca în cazul nostru.

Un sistem ETL proiectat corect extrage date din sistemele sursă, impune standarde de calitate și consistență a datelor, conformează datele astfel încât surse separate să poată fi utilizate împreună și, în cele din urmă, furnizează date într-un format pregătit pentru prezentare, astfel încât dezvoltatorii de aplicații să poată construi aplicații iar utilizatori finali să poată lua decizii (Caserta, 2013).

Modulul de optimizare a transportului vizează optimizarea transportului pentru o companie de transport maritim în termeni de costuri, distanțe, respectiv timp. Cele două cazuri de afaceri se concentrează pe:

- prognozarea locației și orei următoarelor comenzi. În acest caz, camioanele au estimări despre locul în care ar urma următoarea comandă și vor circula în colo, astfel încât atunci când comanda apare, acestea se află în apropiere și preluarea se poate face într-un timp cât mai scurt.

- optimizarea latenței de livrare. Aceasta se referă la timpul de predare, care poate fi întârziat din mai multe motive, cum ar fi rutele suboptimale sau operațiunile sub-optime roll-on / roll-off (RORO) (Sauri, 2012).

Interfața de utilizare multimodală oferă utilizatorului posibilitatea de a configura vizualizarea în funcție de preferințele sale și concentrarea asupra afacerii sale. Datele (respectiv ieșirile modelatorului de date de transport) pot fi afișate de instrumentul de vizualizare a datelor și rezultatele pot fi explicate de explicatorul inteligent al rezultatelor (Gunning, 2017). Arhitectura instrumentului de vizualizare a datelor este explicată în continuare.

3. Abordare

Acest capitol descrie abordarea de verificare și validare în cadrul platformei software Big-Smart-Log. Pentru testarea Big-Smart-Log, se aplică următoarele reguli generale:

- Toate activitățile de testare sunt planificate și urmărite în T1.1 Partajarea și comunicarea datelor;
- Testarea software-ului se face conform ghidului aplicabil descris în secțiunile 4-8;
- Revizuirile se fac în conformitate cu procesul de revizuire aplicabil, și anume evaluarea inter pares de către partenerii individuali în curs de dezvoltare.

3.1 Cartografierea procesului de dezvoltare

Activitățile de dezvoltare și testare ar trebui să se desfășoare în paralel. Activitățile de implementare a testelor ar trebui să înceapă de îndată ce încep activitățile de dezvoltare. În mod clar, testarea reală va începe de îndată ce entitatea de testat este disponibilă.

3.2 Strategia de testare

Atributele de calitate și importanța lor relativă au fost derivate din (Gorton, 2011). Rezultatele sunt raportate în Tabel 1 de mai jos.

Tabel 1 Importanța relativă a atributelor de calitate

Atribut de calitate	Descriere	Importanta relativă (%)
Mentenabilitate	Gradul de eficacitate și eficiență cu care produsul poate fi modificat.	22% (12/53)
Performanță, scalabilitate și capacitate	Performanța în raport cu numărul de resurse utilizate în condițiile menționate.	19% (10/53)
Fiabilitate	Gradul în care un sistem sau o componentă îndeplinește funcții specifice în condiții specificate pentru o perioadă de timp specificată. Include și „Disponibilitate”.	28% (8+7/53)
Securitate	Gradul de protecție a informațiilor și datelor, astfel încât persoanele sau sistemele neautorizate să nu le poată citi sau modifica iar persoanelor sau sistemelor autorizate nu li se refuză accesul.	23% (12/53)
Utilizare	Gradul în care produsul are atribute care îi permit să fie înțeles, învățat, utilizat și atractiv pentru utilizator, atunci când este utilizat în condiții specificate. Include, de asemenea, „Funcționalitate și gestionabilitate”.	8% (4+0/53)

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Există o serie de puncte demne de menționat:
(Gorton, 2011) „Disponibilitate" au fost îmbinate sub „Fiabilitate" pentru a se potrivi cu descrierea atributului de calitate menționată de ISO-25010.

Cerințele nefuncționale din secțiunea „Funcționalitate și gestionabilitate" au fost luate în considerare în „Utilizare", deoarece ISO-25010 nu menționează acest atribut de calitate. În (Gorton, 2011), acest atribut se referă la ușurința utilizării, instalării și gestionării platformei de către utilizatorul final.

Importanța relativă din tabelul de mai sus oferă o indicație a modului de împărțire a efortului de testare peste atributele de calitate.
Importanța relativă a fost determinată pe baza nivelurilor de prioritate atribuite cerințelor nefuncționale. Singurele niveluri luate în considerare sunt Must și Should, întrucât sunt declarate nivelurile care desemnează cerințele care trebuie îndeplinite de platforma implementată. Pentru fiecare atribut de calitate, importanța a fost determinată prin însumarea greutății atribuite (Must = 2, Should = 1, others = 0) pentru nivelul de prioritate atribuit unei cerințe referitoare la acea calitate. Numărul total de puncte din toate categoriile a fost de 53. Acest scor este menit doar ca o valoare pur orientativă.

Exemple ale atributelor de calitate menționate anterior în sistemul prevăzut pot fi găsite în T4.2 în tabelul fiecărei cerințe de la capitolul „Use case".
Atributele de calitate vor fi măsurate folosind Indicele Calității (TQI) al TIOBE, așa cum este descris în secțiunea 8. Testarea nefuncțională, care se bazează și pe ISO-25010. Această potrivire garantează că toate atributele de calitate menționate mai sus sunt gestionate folosind un singur instrument. Facilitatea de funcționare și gestionabilitatea, care nu face parte din ISO-25010, vor fi evaluate utilizând feedback-ul utilizatorului și testele de utilizare.

Deși menținerea nu este o calitate care poate fi testată prin activitățile obișnuite de testare, ea poate fi măsurată, monitorizată și aplicată în mod static. Prin urmare, prin testarea Mentenabilității ne referim la conformitatea valorilor măsurate detectate pe codul sursă cu cadrul de vizualizare și monitorizare din (D5.1).

Pe lângă atributele de calitate menționate în (Gorton, 2011), activitățile de testare includ și teste pentru funcționalitatea adecvată, și anume îndeplinirea cerințelor funcționale și a cazurilor de utilizare. Deoarece eforturile de implementare vor urma prioritatea atribuită într-un astfel de document, testarea va urma aceeași prioritate a testării cerințelor.

3.2.1 Caracteristici de calitate pentru fiecare nivel de test

Testarea bazată pe atribute de calitate nu trebuie făcută pentru toate atributele de calitate de pe fiecare nivel de testare. În acest paragraf, atributele de calitate sunt atribuite unuia sau mai multor niveluri de testare. Pot exista diferite atribute de calitate pentru diferite subsisteme sau module.

Testarea unității
Securitatea ar trebui testată la acest nivel de testare numai în modulele în care codul gestionează datele sensibile ale utilizatorului.

Mentenabilitatea este verificată la nivel de unitate de către partenerii în curs de dezvoltare care trebuie să respecte, individual, cerințele nefuncționale aferente.
Testarea integrității
Performanța ar trebui testată la acest nivel, deoarece ar putea exista mai multe interacțiuni între diferite componente software implementate de același partener sau de diferiți parteneri. Astfel de componente vor fi identificate în arhitectura software, dar conform definițiilor cazurilor

de utilizare, apar următoarele subsisteme: analize extinse bazate pe codul sursă, manipularea și analiza datelor și prognozarea sunt principalul subiect al activităților de testare.

Trebuie testate *funcționalitatea adecvată* la nivel de integrare datorită prezenței mai multor componente arhitecturale care oferă funcționalități cheie prin interacțiunea între ele. Aceste componente pot fi implementate fie de același partener, fie de parteneri diferiți. În ambele cazuri, este necesară testarea la acest nivel.

Testarea sistemului

- *Securitatea* ar trebui să fie acoperită în testele de sistem pentru a evita eventuale scurgeri de date și accesuri neautorizate cauzate de un schimb sensibil de date sensibile între componentele sistemului.
- *Utilizarea* ar trebui acoperită în testele sistemului, deoarece experiența utilizatorului depinde de întregul sistem. Toate interfețele utilizatorului trebuie testate aici.
- *Performanța* ar trebui testată la acest nivel, deoarece performanța generală a sistemelor depinde de mai multe componente care lucrează împreună.
- *Fiabilitatea* trebuie testată la acest nivel de testare, deoarece funcționarea defectuoasă a unei componente poate întrerupe serviciul oferit de întregul sistem.
- *Funcționalitatea adecvată* este testată temeinic la acest nivel pentru a garanta că cerințele cheie (Trebuie să aibă) au fost corect implementate.

Test de admitere

- *Utilizare*, deoarece utilizatorul final trebuie să aprobe ușurința utilizării sistemului final.
- *Funcționalitatea adecvată* este testată în sfârșit și de către utilizatorul final al sistemului, verificând dacă acesta îi îndeplinește așteptările.

Atributele de calitate sunt atribuite nivelului (testelor) de testare în care se potrivesc cel mai bine după cum urmează:

Tabel 2 Atribute de calitate atribuite nivelurilor de testare

Quality Attribute	*Test de unitate*	*Test de integrare*	*Test de sistem*	*Test de acceptare*
Mentenabilitate	+			
Performanţă		+		
Fiabilitate			++	
Securitate	++		+	
Utilizare			+	++
Funcționalitatea adecvată		+	++	+

(Gol) atributul de calitate nu este o problemă la acest nivel;
+ Acest nivel de testare va acoperi acest atribut de calitate;
++ Atributul de calitate va fi acoperit cu atenție - este un obiectiv major la acest nivel de testare.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

4. Testarea principiilor de implementare

Această secțiune descrie principiile de bază de implementare a testelor care stau la baza întregii abordări de testare pentru platforma software Big-Smart-Log. Acest lucru este elaborat în continuare la nivelul corespunzător în fiecare secțiune de specificații de testare.

4.1 Trasabilitate

Trasabilitatea cerințelor testate la fiecare nivel de testare se realizează după cum urmează:

- Trasabilitatea se realizează prin asocierea identificatorilor de caz-test la identificatorii de cerințe folosind matrici dedicate de trasabilitate. Matricea are o coloană pentru ID-ul testului, o coloană pentru ID-ul cerinței funcționale sau nefuncționale și o coloană de descriere dedicată pentru informații suplimentare.
- Matricile vor fi completate după procesul individual de proiectare a testului și fac parte din acest document.

Verificarea dacă cerințele sunt efectiv testate de fiecare test face parte din procesul de revizuire.
Fiecare caz de testare este trasat la cerințele software corespunzătoare, dacă este cazul, și în cele din urmă la cazul de utilizare corespunzător (folosind o coloană suplimentară).
Documentația privind specificațiile testului trebuie să indice ce cerințe software sunt acoperite de fiecare test specificat.

4.2 Activități de testare

Următoarele activități trebuie efectuate:

Planificare și control
Scopul principal al acestei activități este de a oferi îndrumări pentru executarea și testarea activităților de finalizare.

Execuţie
Această activitate constă în principal în executarea articolelor de testare specificate utilizând infrastructura de testare implementată și generarea unui raport asupra rezultatelor.
Înainte de executarea efectivă a testelor, codul va fi compilat automat și verificat pentru controlul calității software-ului. De asemenea, vor fi verificate erorile și avertismentele compilatorului, care nu vor permite avertismentele / erorile L1 și L2.

Executarea testării sistemului și testarea integrării vor face parte din activitatea de integrare continuă. Execuția lor este, prin urmare, complet automatizată.
Cu toate acestea, pentru activitățile de testare unitară, partenerii individuali sunt responsabili de adoptarea propriei strategii de testare atâta timp cât respectă strategia de testare descrisă în secțiunea anterioară.
Prin urmare, partenerii individuali au capacitatea de a specifica individual propria infrastructură de testare și unitățile de testare pe baza arhitecturii propriului instrument. Cu toate acestea, în ceea ce privește raportarea și înregistrarea rezultatelor testării, acestea trebuie să respecte îndrumările menționate mai jos.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

4.2.1 Completare

În această activitate, toate articolele de testare, jurnalele de testare și baza de testare sunt arhivate și se generează un raport de evaluare. Următoarele linii directoare se aplică pentru toate nivelurile de testare:

- Toate execuțiile testului trebuiesc urmărite;
- Acoperirea testului (cod și posibil calea), atunci când este disponibilă, și rezultatele testelor sunt principalele variabile care trebuiesc urmărite;
- Jurnalele de execuție trebuie să fie în format XML / JSON sau ușor convertibile, pentru a facilita generarea automată a raportului și trebuie să includă, pe lângă celelalte informații, ID-ul lor unic de testare;
- Jurnalele de execuție ale testelor trebuie încărcate pe platforma desemnată;
- Instrumentul care va fi utilizat pentru generarea de rapoarte trebuie să accepte XML ca ieșire.

4.2.2 Criterii de intrare

Înainte de începerea testării, trebuie îndeplinite următoarele criterii generale:

- Baza de testare trebuie să fie disponibilă conform descrierii din Tabelul 3.
- Codul trebuie să poată fi construit fără erori de compilare și trebuie să fie disponibil mediul complet pentru a trece de la cod la executabil.
- Pentru testarea statică a documentelor (revizuire) elementele de testare trebuie să fie sub controlul versiunii și în „Starea propunerii interne”;
- Pentru testarea statică a codului, elementele de testare trebuie să fie construibile fără erori de compilare.

Tabel 3 Baza de testare a Big-Smart-Log

ID-ul documentului	Descrierea	Disponibil
D2.3	Arhitectură logistică	30/11/2019
D4.2	Criterii de evaluare și valori pentru algoritmi	31/05/2020
D5.1	Cadrul de vizualizare și monitorizare	31/01/2021

În plus, pentru fiecare nivel de testare, trebuie îndeplinite și următoarele.
Testarea unității

- Livrabilele tehnice din WP2 trebuie să fie cel puțin într-o stare de proiectare avansată.
- Sunt disponibile coduri și unități testabile.
- Mediul de testare este gata.

Testarea integrării

- Testarea unității a fost finalizată cu succes.
- Bugurile cu prioritate maximă găsite în timpul testării unitare trebuie să fie remediate și închise.
- Planul de testare a integrării și mediul de testare pentru testarea integrării sunt gata.
- Livrabilele tehnice din WP2 trebuie să fie în starea lor de versiune finală și livrabilele din WP3 trebuie să fie cel puțin într-o stare de proiectare avansată.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Testarea sistemului

- Testarea integrării a fost finalizată cu succes.
- Bugurile cu prioritate maximă găsite în timpul testării integrării trebuie să fie remediate și închise.
- Livrabilele tehnice de la WP2 și WP3 trebuie să fie în starea lor de versiune finală.
- Sunt definite planuri detaliate de testare a sistemului (folosind piloții WP6 ca bază) și mediul de testare a sistemului este gata.
- Artefactele (adică codul sursă) din testele-pilot definite de sarcinile WP6 sunt disponibile pentru a fi furnizate ca intrare pe platforma Big-Smart-Log pentru testarea sistemului.

Testarea de acceptare

- Testarea sistemului a fost finalizată cu succes, iar mediul de testare a acceptării este gata de implementare.
- Bugurile cu prioritate maximă găsite în timpul testării sistemului trebuie să fi fost remediate și închise.
- Cerințele funcționale și nefuncționale prioritare sunt îndeplinite.
- O versiune beta a sistemului este disponibilă pentru a fi implementată partenerilor furnizori de cazuri de testare.

4.2.3 Criteriul de acceptare

Acest paragraf descrie pentru testul static și dinamic obiectivele pentru a decide dacă un test a trecut sau nu.

Test minim de acoperire

Acoperirea minimă a testelor va fi măsurată cu diferite instrumente, în funcție de platforma de dezvoltare utilizată de partenerul în curs de dezvoltare. În general, pentru proiectele bazate pe Java, JUnit trebuie utilizat ca platformă de testare la nivelul unității. Tabelul 4 prezintă procentele minime de acoperire pe nivel de test.

Tabel 4 Obiective de acoperire pentru acceptare:

Nivelul testului	% Acoperirea codului	% Acoperirea căii	% Acoperirea cerințelor	% Acoperirea cazului testului pilot
Test de unitate	50%	50%	-	-
Test de integrare	-	-	20%	-
Test de sistem	-	-	80%	33% (1/3)
Test de admitere	-	-	-	100% (3/3)

Acoperirea cerințelor va fi măsurată folosind urmele și jurnalele de execuție a testului.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Criterii de acceptare / respingere

Tabelul de mai jos prezintă criteriile dacă un test trece.

Tabel 5 Criterii de acceptare / respingere pentru test de executie

Nivelul testului	Criterii de acceptare / respingere
Test de unitate	Partea codului testat respectă comportamentul așteptat implementat de test.
Test de integrare	Cerința este implementată corect și oferă pe deplin funcționalitatea așteptată în cadrul constrângerii definite de cerințele nefuncționale.
Test de sistem	Cerința este implementată corect și oferă pe deplin funcționalitatea așteptată în cadrul constrângerii definite de cerințele nefuncționale.

4.2.4 Criterii de validare

Criteriile de validare a produselor de lucru livrate și executarea cazurilor de testare a acceptării nu fac parte din acest document. Astfel de activități fac, într-adevăr, parte din task-ul 5.1 și rezultatul aferent livrabilul D5.1.

4.3 Testarea regresiei

Testarea de regresie este activitatea de bază care reduce riscul introducerii de erori în codul sursă existent prin adăugarea de funcționalități, remedierea altor erori sau revizuirea caracteristicilor existente.

Testarea de regresie este de obicei aplicată în etapele avansate de dezvoltare atunci când sistemul a început deja să-și asume o formă și există mai multe funcționalități deja disponibile pentru a fi utilizate. Primele teste de regresie încep în paralel cu activitățile de testare a sistemului.

Strategia de testare a regresiei adoptată în acest proiect este o combinație de testare manuală și automată de regresie. Această alegere permite detectarea tipurilor de bug-uri care nu pot fi detectate doar prin adoptarea unei singure strategii.

4.3.1 Regresia manuală

Strategia de regresie manuală este complet delegată echipelor de dezvoltare, care verifică manual execuția corectă a funcționalităților modificate și a zonelor adiacente. Deoarece aceasta este o activitate care consumă foarte mult timp, este recomandat să fie efectuată numai după efectuarea unor modificări importante care ar putea afecta funcționalitatea de bază a sistemului.

În plus, pentru a verifica codul legat de modificările minore, este bine ca echipele de dezvoltare să pregătească o listă de verificare a funcționalităților minore care trebuie verificate și să le verifice toate împreună o dată.

4.3.2 Regresie automată

Acest tip de testare constă în reexecutarea unui set selectat de teste unitare și de integrare care s-au dovedit a identifica mai multe erori în trecut.

Pentru a selecta un astfel de set, este necesar să se colecteze statistici ale testelor trecute și eșuate în timpul activităților de testare anterioare. Cu toate acestea, dezvoltatorii și testerii pot sugera, de asemenea, teste specifice pe baza experienței lor cu codul și a erorilor anterioare. Testarea de regresie automată este considerată parte a integrării continue, aplicabilă la toate nivelurile de testare.

4.4 Raportarea problemelor

4.4.1 Instrucțiuni

Pentru a menține un flux de lucru eficient pentru remedierea erorilor și pentru a asigura că problemele deschise vor fi rezolvate în timp util, reporterul va urma câteva îndrumări simple.
Înainte de a crea o problemă, se trec prin următoarele instrucțiuni:

- Verificarea Documentației pentru dezvoltatori și Ghidul utilizatorului pentru a asigura că comportamentul pe care se raportează este într-adevăr o eroare, nu o caracteristică;
- Verificarea problemelor existente pentru a asigura că nu se copiază munca cuiva;
- Asigurarea că informațiile pe care urmează să fie raportate sunt o problemă tehnică;
- Dacă sunteți sigur că problema pe care o întâmpinați este cauzată de o eroare, înregistrați o nouă problemă.

4.4.2 Șablon de probleme

Șablonul de raportare a problemelor este un substituent implicit pentru fiecare nouă problemă. Rețineți că un nivel mai ridicat de detaliu din raport crește șansele ca un dezvoltator să poată reproduce problema. Este greu de sfătuit cu privire la orice probleme care nu pot fi replicate.

Titlul problemei

Titlul este o parte vitală a raportului de erori pentru dezvoltator și ajută la identificarea rapidă a unei probleme unice. Un titlu bine scris ar trebui să conțină o explicație clară și scurtă a problemei, punând accent pe cele mai importante puncte.

Descrierea problemei

Condiții prealabile

Descrierea condițiilor prealabile este un început excelent, furnizați informații despre setările de configurare a sistemului pe care le-ați modificat, informații detaliate despre entitățile create (produse, clienți etc.), versiunea Magento. Practic, tot ceea ce ar ajuta dezvoltatorul să creeze același mediu ca și dvs.

Pași pentru a reproduce.

Această parte a raportului de erori este cea mai importantă, deoarece un dezvoltator va folosi aceste informații pentru a reproduce problema. Problema este mai probabil să fie rezolvată dacă poate fi reprodusă. Ar trebui să descrieți cu precizie fiecare pas pe care l-ați făcut pentru a reproduce problema. Trebuie incluse cât mai multe informații, uneori chiar diferențe minore pot fi cruciale.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Rezultatul real și așteptat

Pentru a vă asigura că toți cei implicați în remediere sunt pe aceeași pagină, descrieți cu precizie rezultatul pe care vă așteptați să îl obțineți și rezultatul pe care l-ați observat efectiv după efectuarea pașilor.

Informații suplimentare

Informații suplimentare sunt deseori solicitate atunci când raportul de eroare este procesat, se poate economisi timp oferind jurnale, capturi de ecran, ramură de depozitare și revizuire care a fost verificată pentru a instala Big-Smart-Log sau orice alte artefacte legate de problemă, la judecata testatorului.

5. Testarea unitară

5.1 Scop

Testarea unității este un nivel de testare software în care sunt testate unități individuale/componente ale unui software. Scopul este de a valida faptul că fiecare unitate a software-ului funcționează conform proiectării.

5.2 Domeniul de aplicare

Testarea unitară este primul nivel de testare software și se efectuează înainte de testarea integrării și se concentrează pe codul sursă în sine.

5.3 Abordare

O unitate este cea mai mică parte testabilă a oricărui software. De obicei are una sau câteva intrări și de obicei o singură ieșire. În programarea procedurală, o unitate poate fi un program individual, o funcție, o procedură etc. În programarea orientată obiect, cea mai mică unitate este o metodă, care poate aparține unei clase de bază / super, clasă abstractă sau derivată / clasă copil. Cadrele de testare a unității, driverele, butoanele și obiectele simulate / false sunt utilizate pentru a ajuta la testarea unității. Cadrele de testare unitare vor fi utilizate și în scopuri de integrare continuă, pentru a permite testarea automată de regresie.

Deoarece testarea unitară este strâns legată de dezvoltare, va fi adoptată în procesul de dezvoltare în sine.

6. Testarea integrării

6.1 Scop

Acest capitol are ca scop definirea și descrierea procesului global de testare a integrării care va fi adoptat în cadrul Big-Smart-Log, pe baza selectării unei abordări de testare a integrării și a utilizării mediilor de colaborare și a instrumentelor de integrare continuă care să o susțină.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

6.2 Domeniul de aplicare

Testarea integrării este al doilea nivel de testare efectuat după testarea unitară și înainte de testarea sistemului. În timpul testării integrării, toate unitățile individuale sunt combinate și testate ca un grup pentru a expune defecțiuni în interacțiunea dintre unitățile integrate.

6.3 Abordare

Există mai multe abordări de testare a integrării și cele mai utilizate sunt:

- **Big Bang**: este o abordare a testării integrării în care toate sau majoritatea unităților sunt combinate împreună și testate simultan. Această abordare este urmată atunci când echipa de testare primește întregul software într-un pachet. Testarea integrării Big Bang nu trebuie confundată cu testarea sistemului, deoarece prima testează doar interacțiunile dintre unități, în timp ce cea din urmă testează întregul sistem.
- **De sus în jos**: este o abordare a testării integrării în care unitățile de nivel superior sunt testate primul și unitățile de nivel inferior sunt testate treptat după aceea. Această abordare este urmată atunci când se realizează dezvoltarea de sus în jos. De obicei, unitățile de nivel inferior nu sunt disponibile în fazele inițiale ale dezvoltării, astfel încât Test Stubs sunt folosite pentru a le simula.
- **De jos în sus**: este o abordare a testării integrării în care unitățile de nivel inferior sunt testate mai întâi și unitățile de nivel superior sunt testate treptat după aceea. Această abordare este urmată atunci când se realizează dezvoltarea de jos în sus. De obicei, unitățile de nivel superior nu sunt disponibile în timpul fazelor inițiale ale dezvoltării, astfel încât driverele de testare sunt utilizate pentru a le simula.

Printre abordările de testare a integrării menționate mai sus, abordarea de jos în sus este cea mai potrivită pentru cazul platformei software Big-Smart-Log. Platforma generală este formată din componente individuale (de exemplu, cutii de instrumente), care sunt implementate independent de către partenerii relevanți și vor fi disponibile ca microservicii individuale, unificate în cadrul platformei Big-Smart-Log. Atât abordările de sus în jos, cât și cele de jos în sus se potrivesc bine în strategia de testare a integrării continue, care va fi adoptată pentru implementarea platformei finale, deoarece acestea permit testarea integrării să înceapă în paralel cu dezvoltarea efectivă a platformei. Acestea oferă o flexibilitate mai mare, deoarece componentele individuale sunt integrate în sistemul mai larg de îndată ce sunt disponibile și funcționale, în timp ce comportamentul componentelor care nu sunt încă pregătite este simulat prin intermediul driverelor de testare și al butoanelor, ducând în acest mod la un timp redus la piață. Între cele două abordări ierarhice, abordarea de jos în sus este mai potrivită, deoarece procesul de dezvoltare de jos în sus va fi adoptat pentru implementarea platformei. De fapt, dezvoltarea va începe de la funcționalitățile individuale de nivel scăzut pe care ar trebui să le ofere platforma generală și va progresa cu integrarea treptată a acestor funcționalități în componente și module mai largi.

Înainte de a începe testarea integrării, este important să se asigure că există cel puțin un document de proiectare adecvat disponibil, în care interacțiunile dintre fiecare unitate sunt clar specificate. În livrabilul D2.4 Arhitectura de referință logistică, sunt specificate componentele principale ale sistemului și interfețele dintre ele. În plus, este important ca fiecare unitate separată să fie testată pe unitate înainte de testarea integrării și ca toate testele

să fie automatizate în mod corespunzător în cea mai mare măsură, deoarece testarea manuală poate fi ineficientă deoarece dezvoltatorii trebuie să rețină multe artefacte de construcție și să le testeze manual. Acest lucru poate fi realizat prin activarea integrării continue, adică prin procesul de automatizare a construirii și testării codului de fiecare dată când un membru al echipei face modificări într-un mediu colaborativ.

Integrare continuă

Abordarea generală de integrare a Big-Smart-Log se va baza pe utilizarea unui mediu de colaborare, instrumente de integrare continuă și un plan de versionare. Strategia de mai sus va permite, pe de o parte, tuturor dezvoltatorilor să progreseze cu dezvoltarea propriului modul care lucrează în procese independente, folosind și propriile instrumente de testare și, pe de altă parte, să își integreze modulele între ele în versiunile majore, respectând planul de eliberări prevăzut. Acest lucru va duce, de asemenea, la detectarea deficiențelor la începutul dezvoltării, în care problemele sunt de obicei mai mici și mai ușor de rezolvat.

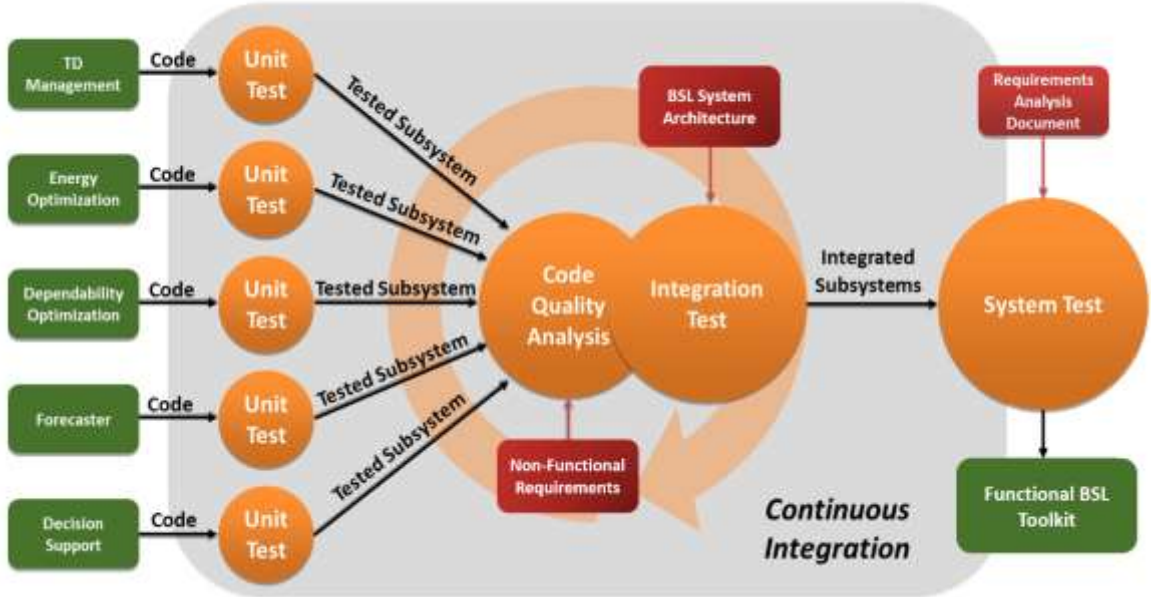


Figura 4 Prezentare generală a procesului de testare Big-Smart-Log

O imagine de ansamblu asupra procesului de testare Big-Smart-Log este ilustrată în Figura 4. În special, Big-Smart-Log va utiliza integrarea continuă pentru a automatiza executarea scripturilor de testare a unității și a integrării incluse ca parte a setului de instrumente iar toate API-urile principale ale modulelor sunt integrate (adică Gestionează TD, Gestionează fiabilitatea, Gestionează consumul de energie, Forecaster și Modulul de asistență pentru decizii). Aceste scripturi efectuează inițial o serie de teste unitare pentru a afirma buna funcționare a fiecărui modul și pentru a se asigura că API-urile funcționează conform așteptărilor. În timpul rulării, ele invocă o serie de componente de testare, fiecare izolată de celelalte, pentru a se asigura că fiecare resursă sau element final funcționează exact așa cum este specificat și documentat. De îndată ce testele unitare au succes, urmează testul de integrare. Toate unitățile individuale (adică module) sunt combinate în conformitate cu abordarea de integrare descrisă mai sus și testate ca un grup pentru a expune defecțiuni

în interacțiunea dintre unitățile integrate, reducând în același timp riscul de noi actualizări care provoacă efecte secundare neașteptate.

Unul dintre cele mai utilizate instrumente pentru integrare continuă este Jenkins. Jenkins este utilizat pentru a construi și testa proiecte software în mod continuu, permițând dezvoltatorilor să integreze cu ușurință modificările proiectelor lor, indiferent de platforma pe care lucrează. Poate fi integrat cu o serie de tehnologii de testare și dezvoltare și este complet configurabil prin intermediul interfeței sale web prietenoase. Cazurile de utilizare tipice ale Jenkins implică crearea unei aplicații dintr-un sistem de control al versiunilor și rularea unei serii de teste automate. Pe de altă parte, prin utilizarea Jenkins se poate realiza testarea imediată a ultimelor modificări, iar dezvoltatorii pot obține feedback imediat cu privire la funcționalitatea codului scris. În cazul în care apare o eroare, codul poate fi readus cu ușurință într-o stare fără erori, fără a pierde prea mult timp pentru depanare.

Activând Jenkins în Big-Smart-Log, executarea testelor va fi declanșată automat de fiecare dată când o modificare a codului este împinsă în managerul de depozitare Git bazat pe web (de exemplu, GitHub, GitLab etc.). De îndată ce execuția testelor este finalizată, pot fi afișate câteva informații utile, cum ar fi numărul de teste care au fost executate, cât a durat executarea și detaliile unui eșec al testului. Cu Jenkins, testarea automată a detaliilor unei anumite defecțiuni poate fi accesată cu ușurință doar făcând clic pe linkul corespunzător. Mai mult, membrii echipei care vor trebui să știe când au fost finalizate testele împreună cu rezultatele corespunzătoare ale testelor pot fi anunțați prin intermediul asistenței Jenkins pentru notificări prin e-mail.

7. Testarea sistemului

7.1 Scop

Acest capitol vizează definirea și descrierea procesului general de testare a sistemului care este adoptat în cadrul Big-Smart-Log, pe baza selectării unei abordări de testare a sistemului și a utilizării parțiale a mediilor de colaborare și a instrumentelor de integrare continuă care să o susțină.

7.2 Domeniul de aplicare

Testarea sistemului este al treilea nivel de testare efectuat după testarea integrării și înainte de testarea acceptării. În timpul testării sistemului, software-ul complet și integrat este testat pentru a verifica dacă cerințele funcționale sunt implementate corect.

7.3 Abordare

Cea mai utilizată abordare pentru testarea sistemului este testarea cutiei negre, cunoscută și sub numele de testare comportamentală. Testarea cutiei negre este o metodă de testare software în care testatorul nu cunoaște structura internă / proiectarea / implementarea elementului testat. Aceste teste pot fi funcționale sau nefuncționale, deși de obicei funcționale.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Figura 5 Testarea cutiei negre

Această metodă este denumită astfel deoarece programul software, în ochii testerului, este ca o cutie neagră; în interiorul căruia nu se poate vedea. Această metodă încearcă să găsească erori în următoarele categorii:

- Funcții incorecte sau lipsă
- Erori de interfață
- Erori în structurile de date sau accesul la baza de date externă
- Comportament sau erori de performanță
- Erori de inițializare și de terminare

Următoarele sunt câteva tehnici care pot fi utilizate pentru proiectarea testelor cutiei negre.

- Partiționare echivalentă: este o tehnică de proiectare a testului software care implică împărțirea valorilor de intrare în partiții valide și nevalide și selectarea valorilor reprezentative din fiecare partiție ca date de testare.
- Analiza valorii limită: este o tehnică de proiectare a testului software care implică determinarea limitelor pentru valorile de intrare și selectarea valorilor care se află la limite și chiar în interiorul / în afara limitelor ca date de testare.
- Graficarea cauzei-efect: este o tehnică de proiectare a testului software care implică identificarea cazurilor (condiții de intrare) și a efectelor (condiții de ieșire), producerea unui grafic cauză-efect și generarea cazurilor de testare în consecință.

8. Testare nefuncțională

8.1 Scop

Scopul testării nefuncționale este de a prelua valori din codul sursă vizat, care oferă o măsură pentru a indica mentenabilitatea, fiabilitatea, compatibilitatea, securitatea și, o oarecare, extindere a adecvării funcționale și a eficienței performanței.

8.2 Domeniul de aplicare

Accentul este pus pe calitatea codului spre deosebire de calitatea cerințelor sau arhitectura. De asemenea, după descrierea valorilor, sunt furnizați indicatori pentru efortul de îmbunătățire a software-ului și sunt descrise și relațiile dintre valori.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

8.3 Fiabilitate și securitate

8.3.1 Acoperirea codului

Pentru a testa funcționalitatea codului sursă, este important ca dezvoltatorii să scrie teste unitare și să se asigure că aceste teste sunt aplicate prin scripturi automate pentru a detecta regresii cât mai curând posibil. Maturitatea testelor unitare poate fi măsurată cu ajutorul „statement coverage” și „branch coverage”.

Aceste valori indică procentul de linii de cod testate și respectiv procentul de ramuri testate în software. Dacă acoperirea testului este redusă, atunci fie unele părți ale codului nu sunt testate deloc, fie unele părți ale codului nu sunt deloc accesibile. TQI ia media dintre „statement coverage” și „branch coverage”.

8.3.2 Interpretare abstractă

Interpretarea abstractă, cunoscută și sub denumirea de „analiza fluxului profund”, este o tehnologie destul de nouă, capabilă să găsească tot felul de erori fatale în software fără a-l rula efectiv. Acest lucru se face prin inspectarea tuturor căilor de execuție posibile prin cod. În acest fel pot fi găsite probleme precum „dereferențele indicatorului nul”, „matricea în afara limitelor”, „împărțirea la zero”, „scurgeri de memorie” și „scurgeri de resurse” (de exemplu, o conexiune la bază de date este deschisă, dar niciodată închisă pentru o anumită cale de execuție). Prin urmare, interpretarea abstractă va acoperi și unele aspecte legate de securitate.
Încălcările detectate ale acestui tip de erori fatale sunt ponderate în funcție de importanța și cantitatea lor. Rezultatul eventual este mapat pe o scară între 0 și 100 în conformitate cu o metodă descrisă în (Steneker2016) . Acesta se numește „factorul de conformitate” sau pe scurt „conformitatea”.

8.3.3 Avertismente ale compilatorului

Majoritatea programelor software trebuie să fie compilatoare înainte de a putea fi executate. Un compilator emite atât erori de compilare, cât și avertismente ale compilatorului în timpul acestui proces. Dacă există erori de compilare într-un program, acesta nu poate fi executat. Pe de altă parte, avertismentele compilatorului nu sunt fatale, dar sunt o indicație importantă dacă există încă probleme importante în software.

Avertismentele compilatorului sunt evaluate în TQI în același mod ca și interpretarea abstractă. Numărul de apariții este luat în considerare împreună cu importanța unui avertisment al compilatorului. Aceasta este conformitatea de avertizare a compilatorului. Rezultatele sunt mapate pe o scară între 0 și 100 în conformitate cu o metodă descrisă în (Steneker2016).

8.4 Testabilitate

8.4.1 Complexitatea ciclomatică

Complexitatea ciclomatică a unei funcții calculează numărul de căi de execuție liniare-independente ale unei funcții așa cum este definită de McCabe [McCabe94]. Aceste valori

sunt utilizate pentru a măsura complexitatea codului și testabilitatea unui sistem software. De obicei, se măsoară complexitatea ciclomatică medie a tuturor funcțiilor. O complexitate medie ciclomatică mai mică de 3 este considerată, în general, ca fiind foarte bună.

8.4.2 Modularitate

Modularitatea unui sistem la nivel de cod este măsurată prin calcularea numărului de dependențe externe per modul, acesta fiind numit și „fan out". În cazul în care numărul mediu de dependențe pe modul este ridicat, devine greu de înțeles sistemul software și de a-l testa izolat. Mai mult, șansele de a reutiliza părți ale sistemului sunt scăzute într-un astfel de caz.

8.5 Mentenabilitate

8.5.1 Standarde de codare

TIOBE definește și menține standardele de codare pentru diferite limbaje de programare pentru clienții săi. Aceste standarde constau din reguli general acceptate la care dezvoltatorii ar trebui să adere pentru a preveni erorile și problemele de întreținere.

Valoarea standard de codare TQI este calculată într-un mod similar, așa cum se face pentru metricele „avertismente ale compilatorului" și „interpretarea abstractă". Pe lângă numărul de încălcări ale standardului, se ia în considerare și gravitatea încălcărilor și dimensiunea sistemului. Rezultatele calculate sunt mapate pe o scară de la 0 la 100 în conformitate cu metoda descrisă în (Steneker2016). Valoarea TQI a acestui factor de conformitate pentru standardele de codare este după cum urmează.

8.5.2 Duplicarea codului

Dacă un sistem software conține o mulțime de coduri similare în diferite locații, atunci acest lucru ar putea influența mentenabilitatea sistemului. Să presupunem că un bug a fost remediat într-o astfel de bucată de cod, atunci există șansa ca bug-ul să nu fie remediat la una dintre locațiile codului duplicat. Codul duplicat are următorul scor TQI. Duplicarea măsurată prin identificarea a 100 de jetoane identice consecutive fără a lua în considerare comentariile și aspectul.

8.5.3 Cod mort

Codul mort într-un sistem software este o risipă inutilă. Costă efort de întreținere. În ciuda faptului că această valoare contează doar pentru o parte foarte mică din calitatea totală a codului, este un bun indiciu al ordonării sistemului.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

8.6 Relații metrice

Pentru a îmbunătăți o anumită valoare, trebuie să depuneți eforturi în activitățile de inginerie software, care vizează această îmbunătățire. Cât de mult efort va costa, nu poate fi descris exact. Ceea ce poate fi descris, totuși, este compararea efortului de a îmbunătăți o anumită valoare. Dacă luăm în considerare influența metricelor unul asupra celuilalt, se poate aplica planificarea strategică.

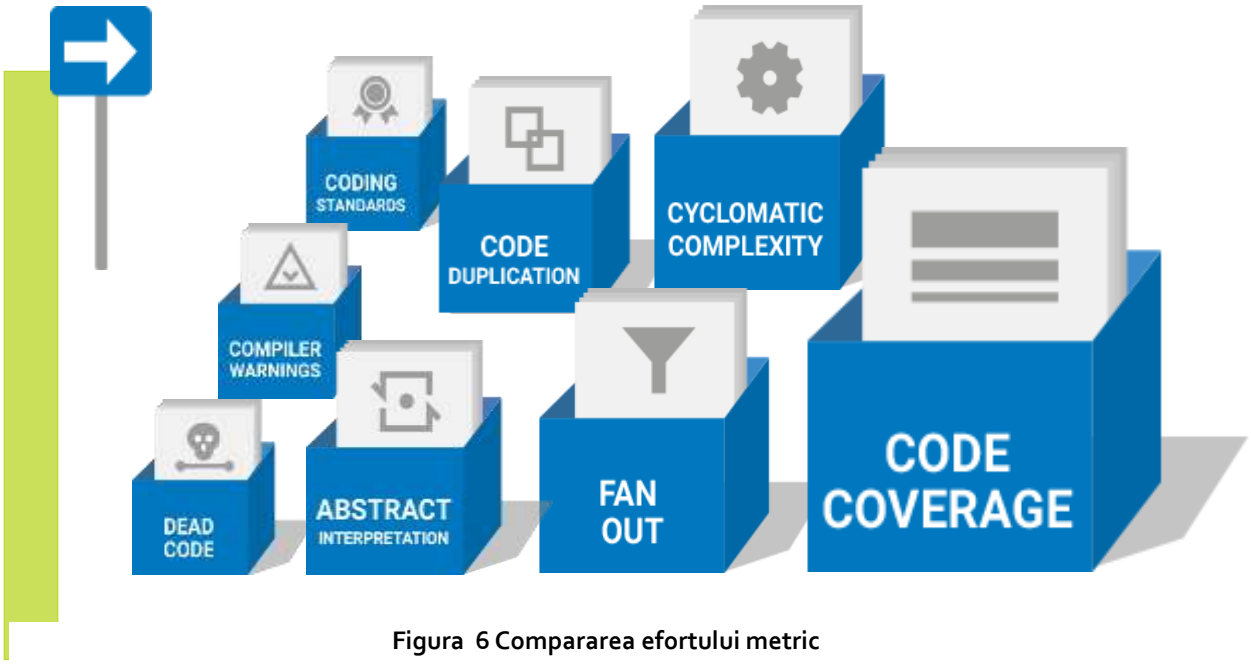


Figura 6 Compararea efortului metric

Imaginea de mai sus indică efortul de a îmbunătăți o valoare. Cu cât este mai mare cutia, cu atât va costa mai mult efort pentru a îmbunătăți acea valoare particulară. Figura 6 arată relația dintre valori. De acolo, schema de prioritizare poate fi recuperată.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

9. Integrare continuă

9.1 Scop

Integrarea continuă se reduce la practica în care dezvoltatorii își îmbină sursele într-un depozit de coduri. Un sistem de asamblare construiește apoi sursele și cadrele de testare, și își execută testele disponibile. Efectuarea manuală a acestor pași este laborioasă și greoaie. Cu toate acestea, prin automatizarea acestui proces, devine foarte puternic, deoarece rezultatele de construcție și testare sunt disponibile rapid și sunt create în mod constant.

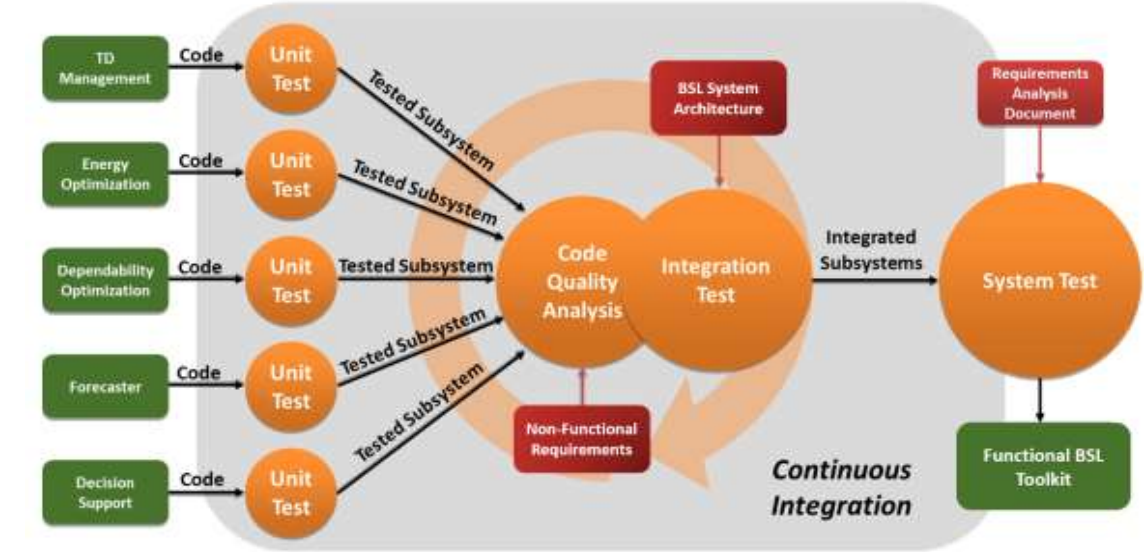


Figura 7 Prezentare generală a procesului de testare Big-Smart-Log

9.2 Practica obișnuită

Primul pas este menținerea unui depozit comun. Fiecare componentă a Big-Smart-Log va avea propriul său substituent în depozit, unde fiecare dezvoltator își poate aloca sursele sau așa-numitele artefacte. Acest lucru este necesar, astfel încât un instrument de integrare continuă (CI) poate prelua aceste surse pentru a le furniza unui sistem de construire.

Al doilea pas este de a avea unul sau mai multe sisteme de construire, unul pentru fiecare componentă. Aceste sisteme de construcție vor fi furnizate cu sursele, preluate din depozit de către CI. Construirea va fi invocată și de CI. Apoi, CI poate verifica, analizând eroarea standard sau eroarea standard, dacă versiunea a reușit sau nu.

În cele din urmă, al treilea pas este utilizat pentru a conecta cadre de testare și calitate. CI va invoca cadrele configurate, astfel încât fiecare cadru să își poată îndeplini datoria. CI poate colecta, de asemenea, rezultatele din aceste cadre în scopuri de prezentare și raportare.

The steps above are repeated at least once a day. It is common practice to apply an incremental approach to speed-up the process and have quick feedback.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

10. Metodologia de validare

Această secțiune prezintă metodologia de validare, care va asigura că modelul cazului de afaceri este adecvat pentru testarea platformei BIG-SMART-LOG pe baza cerințelor generale definite de EKOL.

Fluxul de validare

Validarea modelului software urmează fluxul descris în Figura 8.

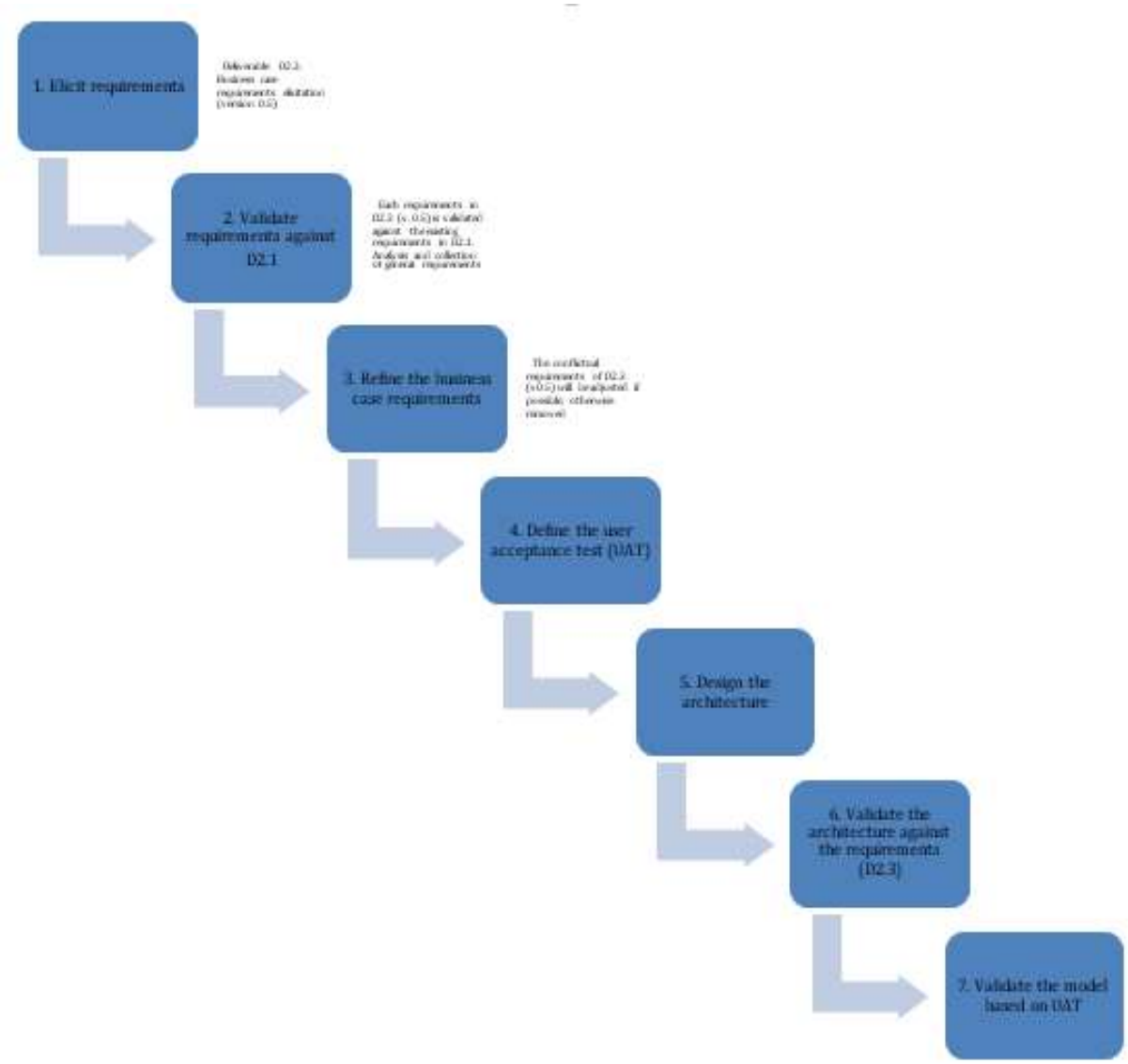


Figura 8 Metodologia de validare

10.1 Definirea cerințelor

Cerințele cazului de afaceri (BC) sunt eliberate de la Ekol. Versiunile finale ale cerințelor, specificațiilor și cadrului Big-Smart-Log fac obiectul Milestone M2 și sarcinilor WP2 în care sunt specificatr definițiile problemei companiei Ekol.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

10.2. Validarea cerințelor

Cerințele BC sunt validate în funcție de cerințele funcționale și nefuncționale ale platformei Big-Smart-Log, subiectul sarcinii *T4.2. Analiza și colectarea cerințelor generale*. Pe baza acestei validări, apar trei cazuri:

- Cerințele BC sunt valabile;
- Cerințele BC sunt în afara scopului proiectului;
- Cerințele BC sunt în conflict cu cerințele generale.

10.3. Rafinarea cerințelor BC

Cerințele BC sunt tratate în funcție de tipul lor, așa cum se arată în **Error! Reference source not found.**Tabel 6.

Tabel 6 Rafinarea cerințelor BC

Tipul cerinței BC	Rafinament
Cerință valabilă	-
În afara domeniului de aplicare	Acceptat ca atare
Cerință conflictuală	a) Dacă este posibil, reglați pentru a se potrivi cerințelor generale b) În caz contrar, scoateți-l

Ori de câte ori este descoperită o cerință conflictuală, aceasta trebuie atenuată fie prin adaptarea acesteia la cerințele generale, fie prin eliminarea acesteia. Ambele cazuri, proprietarul acesteia este informat și ea este cea care decide măsurile adecvate de atenuare.

10.4. Definirea testelor de acceptare a utilizatorului (UAT)

Testarea acceptării utilizatorului (UAT) constă într-un proces de verificare a faptului că o soluție funcționează pentru utilizatorul care proiectează arhitectura (Hambling, 2013).

10.5. Validarea arhitecturii în funcție de cerințe

Este disponibilă o suită de trei metode, toate dezvoltate la Institutul de Inginerie Software, și anume:

- Procesul ATAM (Architecture Tradeoff Analysis Method) constă în adunarea părților interesate pentru a analiza factorii de afaceri (funcționalitatea sistemului, obiectivele, constrângerile, proprietățile nefuncționale dorite) și din acești factori extrag atribute de

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

calitate care sunt utilizate pentru a crea scenarii. Aceste scenarii sunt apoi utilizate împreună cu abordările arhitecturale și deciziile arhitecturale pentru a crea o analiză a compromisurilor, a punctelor de sensibilitate și a riscurilor (sau non-riscurilor). Această analiză poate fi convertită în teme de risc și impactul acestora, după care procesul poate fi repetat. Cu fiecare ciclu de analiză, procesul de analiză continuă de la mai general la mai specific, examinând întrebările care au fost descoperite în ciclul anterior, până când arhitectura a fost reglată și temele de risc au fost abordate (Kazman , 1998).

- SAAM (Software Architecture Analysis Method): este o metodă utilizată în arhitectura software pentru a evalua o arhitectură de sistem. A fost prima metodă de analiză a arhitecturii software documentată și a fost dezvoltată la mijlocul anilor 1990 pentru a analiza un sistem de modificabilitate, dar este util pentru testarea oricărui aspect nefuncțional (Dobrica, 2002).
- ARID (Active Reviews for Intermediate Designs) este o metodă de evaluare a arhitecturilor software care combină aspecte din metoda de analiză a arhitecturii (ATAM) și metoda de analiză a arhitecturii software (SAAM) într-un nivel mai tactic (Clements, 2000).

10.6. Validarea modelul pe baza UAT

Criteriile testului de acceptare a utilizatorului (UAT) (în dezvoltarea software-ului agil) sunt de obicei create de clienți de afaceri și exprimate într-un limbaj de domeniu de afaceri (Cimperman, 2006). Acestea sunt teste la nivel înalt pentru a verifica exhaustivitatea unei povești de utilizator sau a unor povești „redate” în timpul oricărui sprint / iterație.

11. Metodologia de testare

11.1. Analiza codului

Metodologia de testare include un pas numit Analiza codului, un pas important pentru a limita creșterea datoriei tehnice. Importanța acestui pas este fundamentală, deoarece inginerii se concentrează de obicei pe optimizările de cod din punct de vedere al eficienței, și nu din cel de întreținere. Aceasta înseamnă că, în viitor, codul tinde să fie dezorganizat, greu de înțeles și de înțeles, greu de depanat și reparat. Platforma Big-Smart-Log, dezvoltată în platforma software Big-Smart-Log (în care HOLISUN este partener), oferă exact acest tip de analiză și sugestii pentru o bază de cod mai bună. O altă piesă software utilizată în acest domeniu este SonarQube.

În primul rând, am explorat analiza evoluției și caracteristica de evaluare TD a setului de instrumente TD. Evoluția datoriei tehnice de-a lungul versiunilor software-ului este prezentată în Figura 9. Aspectele TD care sunt surprinse și pe care le-am monitorizat sunt: dobânda (EUR), principalul (EUR) punctul de rupere și dobânda cumulată (EUR)). Sumele sunt exprimate în EUR, ca o estimare a **costului de timp * pe unitate de timp** care ar necesita eliminarea completă a TD. Deci, aceasta înseamnă pierderea efectivă a proiectului.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

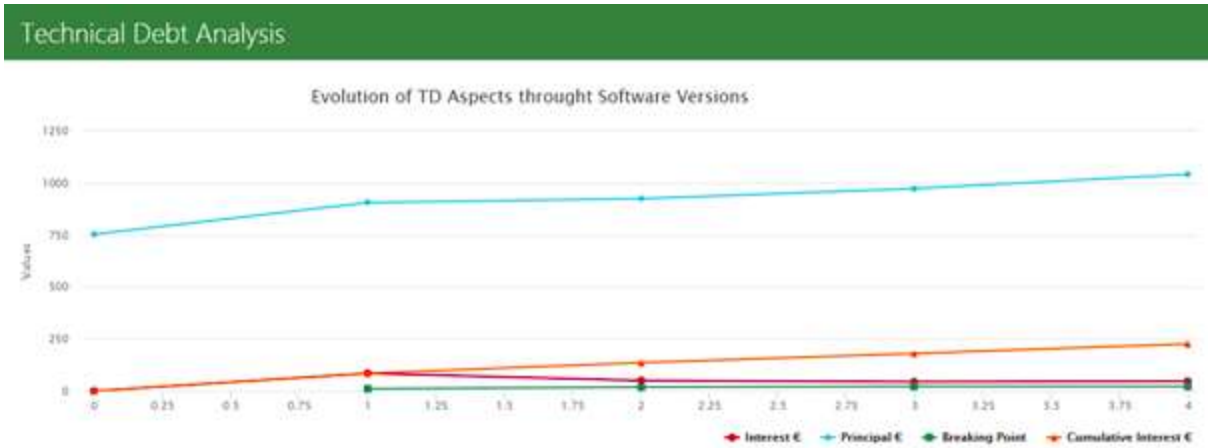


Figura 9 Evoluția TD pe parcursul versiunilor

După cum se poate vedea în Figura 9, TD crește orele suplimentare (atât principalul, cât și dobânda cumulată), cu toate acestea dobânda (linia roșie) merge în jos. Se așteaptă creșterea continuă a principalului TD și dobânda cumulată, deoarece baza de cod a aplicației devine din ce în ce mai mare. Pe partea pozitivă, interesul scăzut sugerează ca echipa de dezvoltare să mențină costurile de întreținere la niveluri bune, asigurând sustenabilitatea sistemului. Rezumatul proiectului în ceea ce privește TD (Figura 10), arată că:

● Principalul TD însumează până la 1364 minute (adică 22,73 ore) sau 909,33 EUR. Acest lucru va crește în timp dacă nu se iau măsuri. TD este foarte evident atunci când trebuie întreprinse acțiuni rapide pentru menținerea software-ului la fața locului, sub presiunea timpului.

● În cod existau 13 vulnerabilități potențiale, au fost identificate 191 mirosuri de cod și 30 de duplicări; cu toate acestea, nu au fost observate erori în timpul analizei.

Figura 10 Rezumatul TD în ceea ce privește platforma Big-Smart-Log

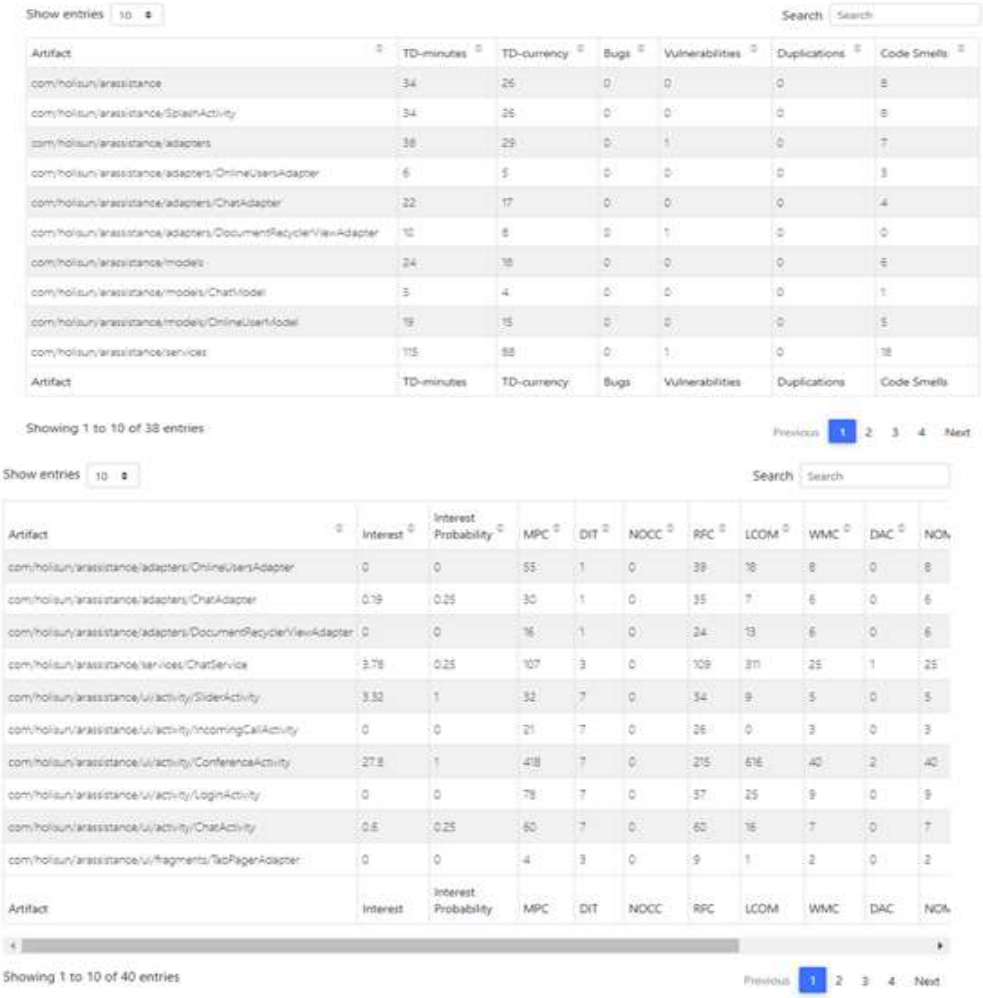
Rezumatul proiectului în termeni de interes TD (Figura 11) arată că:

- Punctul de rupere va fi atins la versiunea 22 a software-ului. Aceasta înseamnă că, având în vedere condițiile actuale, în versiunea 22, dobânda cumulată va fi mai mare decât principalul corespunzător. Cu toate acestea, deoarece proiectul se află în prezent în versiunea 4 (pe baza analizei), punctul de rupere se află cu mult în viitor. Cu toate acestea, practicile de monitorizare continuă privind controlul calității trebuie menținute la niveluri ridicate

- Dobânda totală este de 46,74 EUR cu o probabilitate de 13,89%. Acest lucru înseamnă că nu se rezumă prea multe dobânzi, iar stabilirea datoriei tehnice nu ar trebui să fie de mare prioritate în companie; cu toate acestea, trebuie luate măsuri înainte ca suma să crească prea mult (de exemplu, un salariu lunar);

Figura 11 Rezumatul proiectului de interes

În cele din urmă, o analiză detaliată a artefactelor, așa cum se arată în Figura 12, sugerează că pot fi evidențiate hotspoturi de proiectare în baza de cod, cum ar fi clasa ConferenceActivity, care concentrează sume mari de dobândă și se modifică destul de frecvent.



Artifact	TD-minutes	TD-currency	Bugs	Vulnerabilities	Duplications	Code Smells
com/holisun/arassistance	34	26	0	0	0	8
com/holisun/arassistance/SplashActivity	34	26	0	0	0	8
com/holisun/arassistance/adapters	38	29	0	1	0	7
com/holisun/arassistance/adapters/OnlineUsersAdapter	6	5	0	0	0	3
com/holisun/arassistance/adapters/ChatAdapter	22	17	0	0	0	4
com/holisun/arassistance/adapters/DocumentRecyclerViewAdapter	10	8	0	1	0	0
com/holisun/arassistance/models	24	18	0	0	0	6
com/holisun/arassistance/models/ChatModel	5	4	0	0	0	1
com/holisun/arassistance/models/OnlineUserModel	19	15	0	0	0	5
com/holisun/arassistance/services	115	88	0	1	0	18
Artifact	TD-minutes	TD-currency	Bugs	Vulnerabilities	Duplications	Code Smells

Showing 1 to 10 of 38 entries

Artifact	Interest	Interest Probability	MPC	DIT	NOCC	RFC	LCOM	WMC	DAC	NOM
com/holisun/arassistance/adapters/OnlineUsersAdapter	0	0	55	1	0	39	18	8	0	8
com/holisun/arassistance/adapters/ChatAdapter	0.19	0.25	30	1	0	35	7	6	0	6
com/holisun/arassistance/adapters/DocumentRecyclerViewAdapter	0	0	16	1	0	24	13	6	0	6
com/holisun/arassistance/services/ChatService	3.78	0.25	107	3	0	109	311	25	1	25
com/holisun/arassistance/ui/activity/SliderActivity	3.32	1	32	7	0	34	9	5	0	5
com/holisun/arassistance/ui/activity/IncomingCallActivity	0	0	21	7	0	26	0	3	0	3
com/holisun/arassistance/ui/activity/ConferenceActivity	27.8	1	418	7	0	215	616	40	2	40
com/holisun/arassistance/ui/activity/LoginActivity	0	0	78	7	0	57	25	9	0	9
com/holisun/arassistance/ui/activity/ChatActivity	0.6	0.25	60	7	0	60	16	7	0	7
com/holisun/arassistance/ui/fragments/TabPagerAdapter	0	0	4	3	0	9	1	2	0	2
Artifact	Interest	Interest Probability	MPC	DIT	NOCC	RFC	LCOM	WMC	DAC	NOM

Showing 1 to 10 of 40 entries

Figura 12 Analiza interesului pe clasă

Trecând evaluarea noastră la mecanismele de reducere a TD, am reușit să obținem indicații asupra unor părți ale codului, unde am putea aplica refactorizarea (cum ar fi divizarea metodei

lungi sau mutarea claselor între pachete) sau indicii pentru încălcările obișnuite ale calității, așa cum este prezentat în Figura. 13.

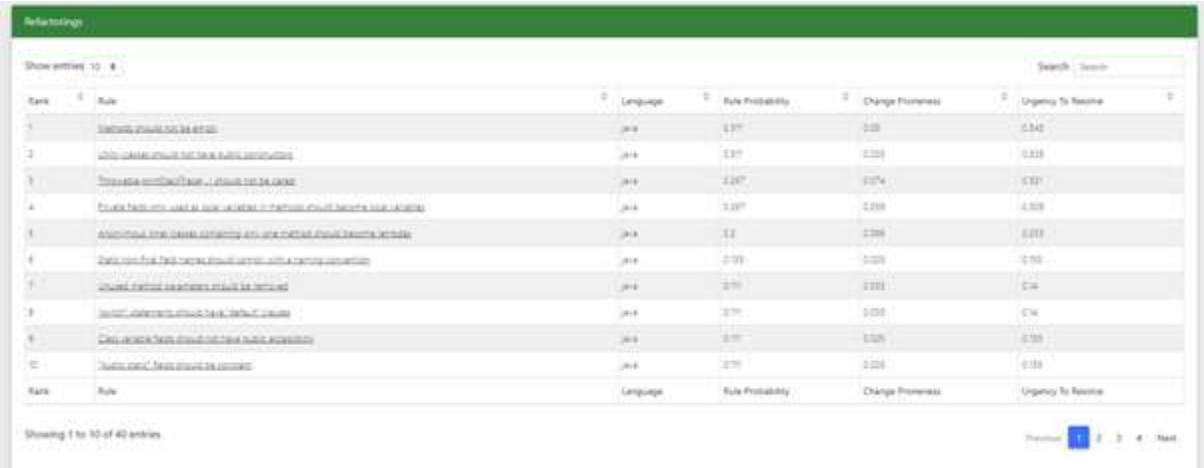


Figura 13 Regulile frecvent încălcate

11.2 Setul de instrumente pentru fiabilitate

Fiabilitatea este evaluată în ceea ce privește securitatea și punctul de control optim.

11.2.1 Securitate

Indicele de securitate al codului sursă al cazului de utilizare auto este de 72% (adică 4 stele din 5). Valoarea se datorează faptului că aplicația este în stadii incipiente, tehnologia nu este încă matură și se schimbă foarte des. Obiectivul nostru este să fie peste 75%, de preferință peste 85%, prin urmare codul este refactorizat având în vedere problemele de securitate raportate de instrument.

Figura 14 Indicele de securitate

Scorurile caracteristicilor, care se referă la confidențialitate, disponibilitate și integritate (Figura 15) sunt:

- **Măsurile de confidențialitate** (0.8) protejează informațiile împotriva accesului neautorizat și a utilizării necorespunzătoare. Majoritatea sistemelor informaționale găzduiesc informații care au un anumit grad de sensibilitate. Ar putea fi informații comerciale proprietare pe care concurenții le-ar putea folosi în avantajul lor sau informații personale cu privire la angajații, clienții sau clienții unei organizații. Acest lucru este peste standardul industriei.

- **Măsurile de disponibilitate** (0.7) protejează accesul la timp și neîntrerupt la sistem. Unele dintre cele mai fundamentale amenințări la adresa disponibilității sunt de natură

non-rău intenționată și includ eșecuri hardware, timp de nefuncționare neprogramat al software-ului și probleme cu lățimea de bandă a rețelei. Atacurile rău intenționate includ diverse forme de sabotaj menite să provoace daune unei organizații prin refuzarea accesului utilizatorilor la sistemul de informații.

- **Măsurile de integritate** (0,7) protejează informațiile împotriva modificărilor neautorizate. Aceste măsuri oferă asigurarea exactității și exhaustivității datelor. Necesitatea protejării informațiilor include atât datele stocate pe sisteme, cât și datele transmise între sisteme, cum ar fi e-mailul. În menținerea integrității, nu este necesar doar să se controleze accesul la nivel de sistem, ci să se asigure în continuare că utilizatorii sistemului pot modifica doar informațiile pe care sunt legitimați să le modifice.

În aplicații similare, confidențialitatea, disponibilitatea și integritatea sunt, în general, acceptate la 75%, ceea ce înseamnă că, în cazul nostru specific, confidențialitatea este bună, dar trebuie să se lucreze la disponibilitate și integritate.

Figura 15 Scoruri caracteristice

Scorurile proprietăților (Figura 16) sunt rezumate în Tabelul 7. Scorurile proprietăților se referă la instrucțiuni sensibile care se ocupă de excepții, memorie sau alte resurse:

Tabel 7 Scorul proprietăților

Proprietate	**Scorul**	**Aparențe**
Manipularea resurselor	80%	385
Misiune	85%	447
Manevrarea excepțiilor	80%	14

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Funcționalitate utilizată greșit	72%	86
Sincronizare	20%	16
Indicator nul	85%	6
Complexitate	82%	N/A
Coeziune	88%	N/A
Cuplare	50%	N/A
Incapsularea	90%	N/A

Resursele sunt gestionate în 385 de cazuri, există 447 de sarcini și 14 gestionări de excepție. Instrumentul a identificat 86 de funcționalități folosite greșit, 16 sincronizări și 6 posibile pointeri nuli. Coeziunea este o măsură a complexității unei singure unități de program, iar cuplarea este o măsură a complexității relațiilor sau legăturilor dintre unitățile de program. Cifrele arată că lucrările sunt în desfășurare și se îmbunătățesc. Cu toate acestea, toate aceste probleme trebuie tratate cu atenție.

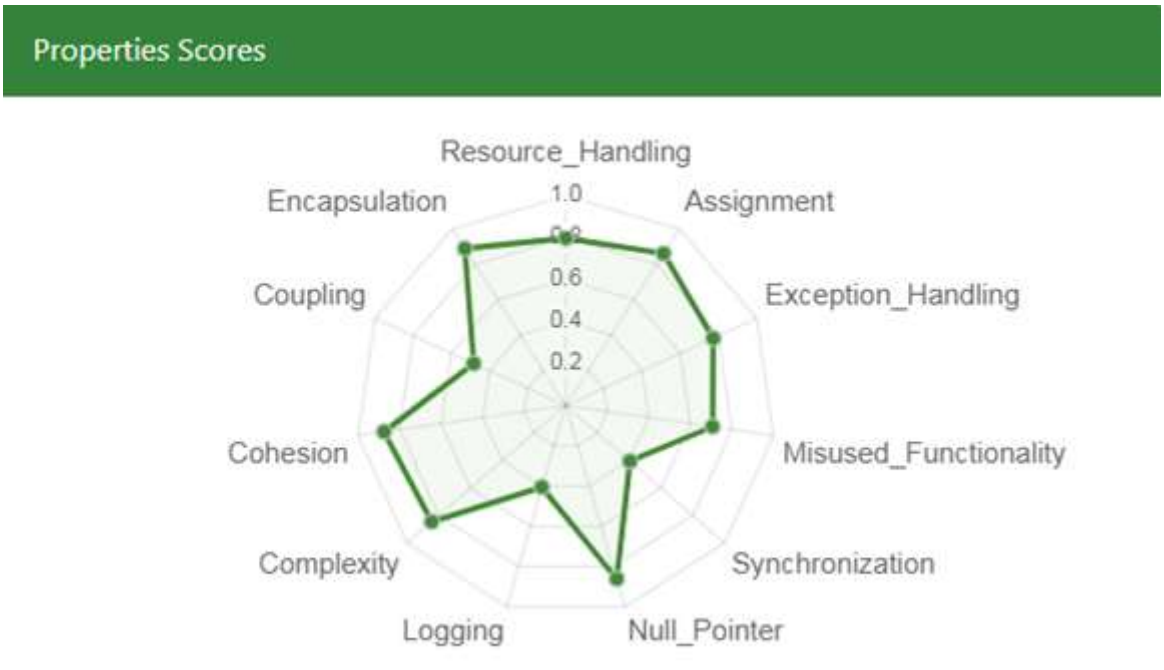


Figura 16 Scorul proprietăților

Harta de căldură de predicție a vulnerabilității (vezi Figura 17) variază clasele de la 0 la 1, cu toate acestea, media este de aproximativ 0,5. Un hotspot de securitate evidențiază o bucată

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

de cod sensibilă la securitate pe care dezvoltatorul trebuie să o revizuiască. După examinare, veți găsi fie că nu există nici o amenințare sau trebuie să aplicați o soluție pentru a securiza codul. Clasele cu nivel zero înseamnă că nu are vulnerabilități de securitate detectate, în timp ce clasele cu nivelul 1 sunt foarte vulnerabile și trebuie reproiectate și rescrise imediat.

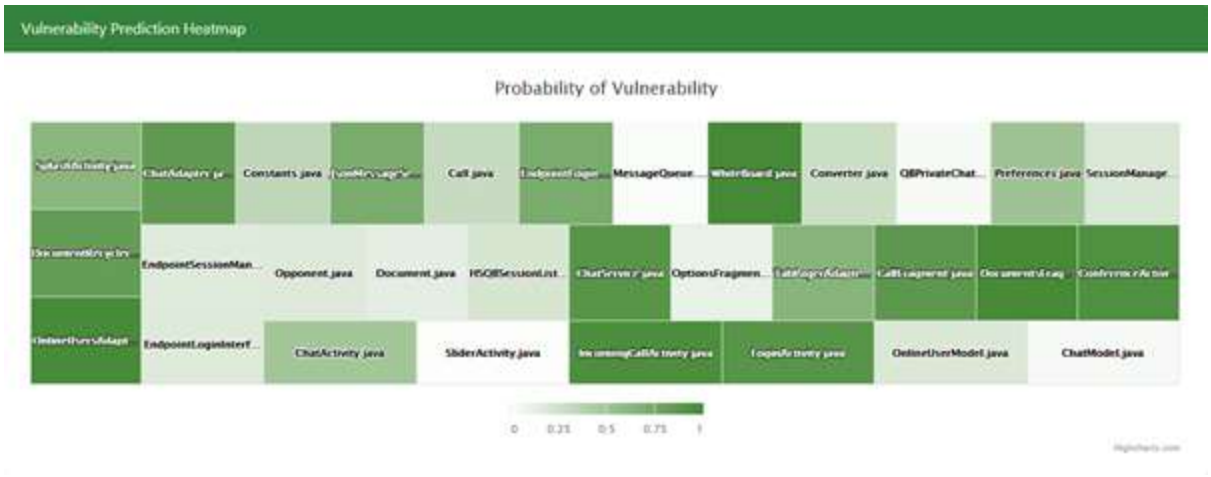


Figura 17 Harta termică de predicție a vulnerabilității

Rezultatele predicției vulnerabilității arată 31 de intrări (a se vedea Figura 18). O vulnerabilitate este un punct din cod care este deschis atacului. Dacă probabilitatea de vulnerabilitate este peste 0,5, clasa este considerată vulnerabilă. Clasele de risc sunt cele care se ocupă de comunicarea între punctele finale, respectiv cele care gestionează interacțiunile utilizatorului. Au fost luate măsuri pentru monitorizarea vulnerabilităților, cum ar fi: validarea datelor și resurselor utilizatorilor, respectiv securizarea comunicării.

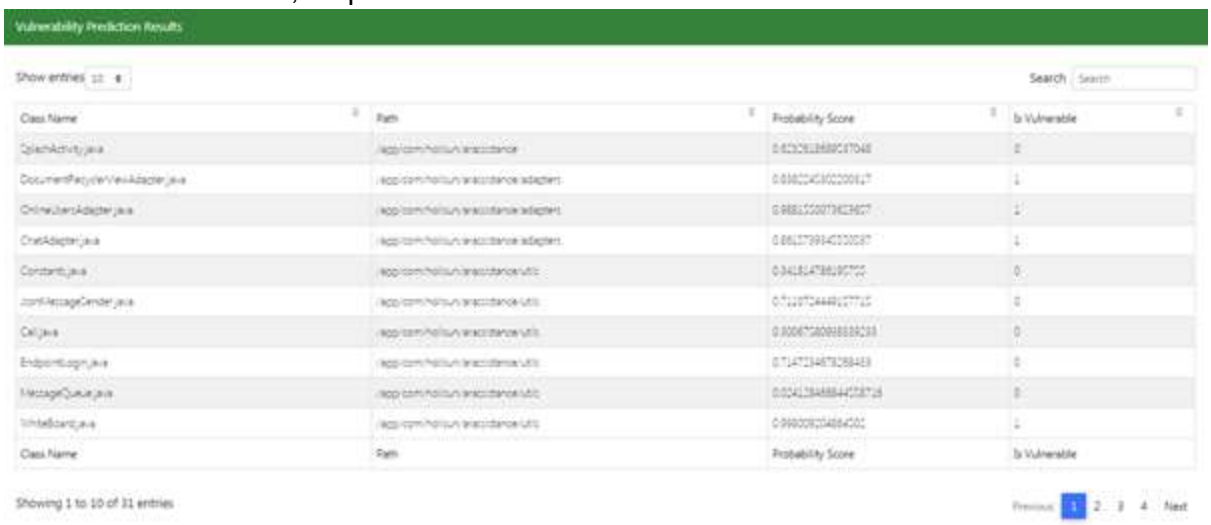


Figura 18 Rezultatul prezicerii vulnerabilității

11.2.2 Punct de control optim

Punctele de control sunt utilizate pentru a asigura fiabilitatea executării programelor. Considerăm consumul de energie pentru execuția programului, în plus față de durata de funcționare a acestuia, drept criterii care trebuie utilizate pentru a minimiza costul general datorat punctelor de control. Sunt derivate expresii noi atât pentru timpul de rulare al programului, cât și pentru consumul de energie corespunzător, care includ probabilitatea de

eșec pe fiecare execuție a instrucțiunilor și cheltuielile generale suportate pentru fiecare punct de control. Figura 19 arată numărul minim de instrucțiuni între punctele de control este 1999. Timpul de execuție crește foarte lent. Aceasta înseamnă că punctele de control pot fi setate la un număr mai mare de instrucțiuni (timpi de execuție).

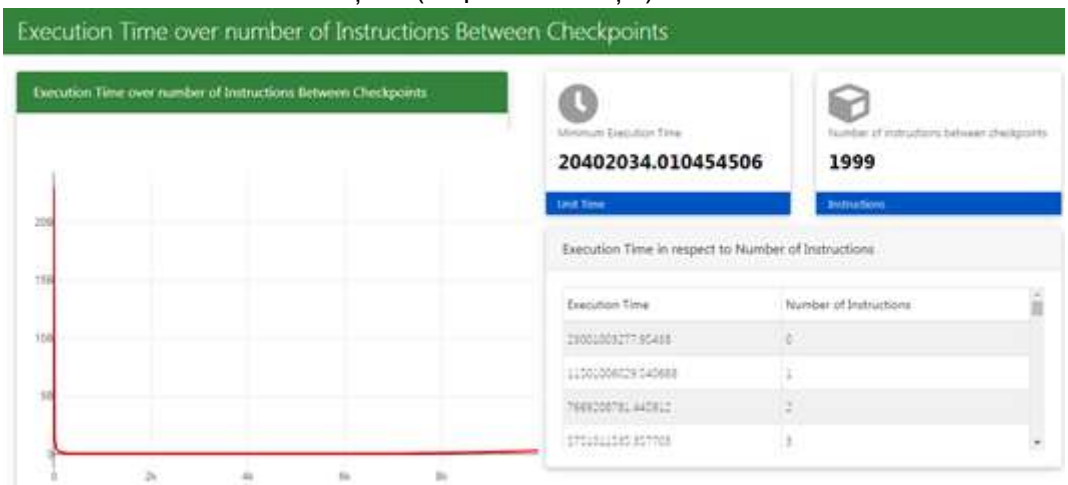


Figura 19 Timpul de execuție peste numărul de instrucțiuni între punctele de control

Figura 20 arată că consumul de energie crește foarte abrupt. Consumul minim este 225200.354, iar numărul de instrucțiuni între punctele de control este de 42. Aceasta înseamnă că fiecare instrucțiune este critică pentru consumul de energie. Am luat decizia de a reduce numărul de instrucțiuni și de a le limita la cele esențiale (de exemplu, eliminați verificările inutile, scrieți un cod de nivel inferior, reduceți utilizarea funcțiilor încorporate și rescrieți-le).

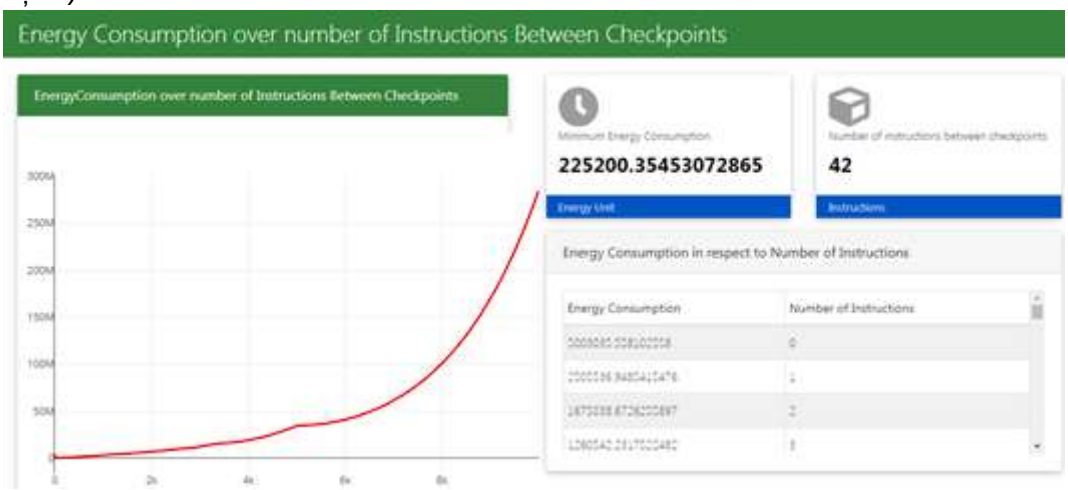


Figura 20 Consumul de energie peste numărul de instrucțiuni între punctele de control

11.2.3 Analiza rezultatelor

Pe baza configurării de mai sus, platforma Big-Smart-Log, dezvoltată de Holisun, a fost analizată în Cutia de instrumente Big-Smart-Log. Au fost adunate următoarele date:

- Utilizarea memoriei (octeți)
- Încărcare CPU (procent)
- Frecvența procesorului (Hertz)

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

• Comutatoare de context (număr)
• Durata cazului de testare (secunde)
• Apeluri de sistem (număr)
• Energie totală (Millijoules)

Setul complet al acestor puncte de date este suficient pentru a caracteriza software-ul executat în ceea ce privește eficiența energetică a acestuia. Figura 21 prezintă măsurătorile exacte extrase.

Figura 21 Analiza energetică a platformei Big-Smart-Log

De asemenea, prezentăm datele sub formă de tabel, pentru o mai bună lizibilitate Tabelul 8. Unele observații notabile cu privire la rezultatele obținute sunt:

1. Încărcarea CPU este o medie a seriei temporale respective. Rețineți că aplicația nu a fost executată izolat total și, prin urmare, datele sunt supuse aliasării de către restul contextului platformei.
2. Împărțind energia totală cu timpul de execuție al testului, obținem o citire medie a puterii de 12,39 / 5,93 = ~ 2,1 W.
3. Cazul de testare nu a subliniat procesorul cu mai mult de 50% din capacitatea sa.

Tabel 8 Analiza energiei sub formă de tabel

Memorie (octeți)	Încarcare CPU (%)	Frecvența CPU (kHz)	Comutatoare de context (#)	Durata cazului de test (sec)	Apeluri sistem (#)	Energia Totală (Joules)
1733098	49.95	1374600	23604	5.93	278	12.39

După finalizarea cu succes a analizei codului, împreună cu ajustările necesare, am trecut la testarea efectivă a platformei, conform metodologiei.

11.3 Testare funcțională vs. nefuncțională

Scopul utilizării a numeroase metodologii de testare în procesul de dezvoltare este de a vă asigura că software-ul poate funcționa cu succes în medii multiple și pe diferite platforme. Acestea pot fi de obicei împărțite între testarea funcțională și cea nefuncțională. Testarea funcțională implică testarea aplicației în funcție de cerințele afacerii. Incorporează toate tipurile

de testare concepute pentru a garanta că fiecare parte a unui software se comportă conform așteptărilor prin utilizarea cazurilor de utilizare furnizate de echipa de proiectare, de analistul de afaceri sau, în cazul nostru, de furnizorul de cazuri de utilizare care este Ekol.

Aceste metode de testare sunt de obicei realizate în ordine și includ:

- Testarea unității
- Testarea integrării
- Testarea sistemului
- Testarea acceptării

Metodele de testare nefuncționale încorporează toate tipurile de testare axate pe aspectele operaționale ale unui software. Acestea includ:

- Testarea performanței
- Testarea securității
- Testarea utilizabilității
- Testarea compatibilității

Cheia pentru lansarea unui software de înaltă calitate care poate fi adoptat cu ușurință de către utilizatorii finali este utilizarea unui cadru de testare robust care să implementeze atât metodologii de testare software funcționale, cât și nefuncționale.

➢ Testarea unității

Testarea unitară este primul nivel de testare și este adesea efectuată chiar de dezvoltatori. Este procesul de asigurare a componentelor individuale ale unui software la nivel de cod, funcționale și funcționale așa cum au fost concepute. Dezvoltatorii într-un mediu bazat pe test vor scrie și vor rula testele înainte ca software-ul sau caracteristica să fie transmise echipei de testare. Testarea unității poate fi efectuată manual, dar automatizarea procesului va accelera ciclurile de livrare și va extinde acoperirea testelor. Testarea unității va facilita, de asemenea, depanarea, deoarece găsirea problemelor mai devreme înseamnă că necesită mai puțin timp pentru a remedia decât dacă ar fi descoperite mai târziu în procesul de testare.

➢ Testarea integrării

După ce fiecare unitate este testată temeinic, este integrată cu alte unități pentru a crea module sau componente care sunt proiectate pentru a îndeplini sarcini sau activități specifice. Acestea sunt apoi testate ca grup prin teste de integrare pentru a se asigura că segmente întregi ale unei aplicații se comportă așa cum era de așteptat (de exemplu, interacțiunile dintre unități sunt perfecte). Aceste teste sunt adesea încadrate de scenarii de utilizator, cum ar fi logarea într-o aplicație sau deschiderea fișierelor. Testele integrate pot fi realizate fie de dezvoltatori, fie de testeri independenți și constau de obicei dintr-o combinație de teste funcționale și manuale automatizate.

➢ Testarea sistemului

Testarea sistemului este o metodă de testare a cutiei negre utilizată pentru a evalua sistemul completat și integrat, ca întreg, pentru a se asigura că îndeplinește cerințele specificate. Funcționalitatea software-ului este testată de la capăt la cap și este de obicei condusă de o echipă de testare separată decât echipa de dezvoltare înainte ca produsul să fie introdus în producție.

➢ Testarea acceptării

Testarea acceptării este ultima fază a testării funcționale și este utilizată pentru a evalua dacă platforma finală este sau nu gata pentru livrare. Aceasta implică asigurarea conformității produsului cu toate criteriile comerciale inițiale și cu satisfacerea nevoilor utilizatorului final.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Acest lucru necesită ca produsul să fie testat atât intern, cât și extern, ceea ce înseamnă că va trebui să îl puneți în mâinile utilizatorilor finali pentru testarea beta, împreună cu cei ai echipei dvs. QA. Testarea beta este esențială pentru a obține feedback real de la potențialii clienți și poate rezolva orice problemă finală de utilizare.

- Testarea performanței

Testarea performanței este o tehnică de testare nefuncțională utilizată pentru a determina cum se va comporta o aplicație în diferite condiții. Scopul este de a-i testa capacitatea de răspuns și stabilitatea în situații reale ale utilizatorilor. Testarea performanței poate fi împărțită în patru tipuri:

- **Testarea încărcării** este procesul de punere a unei cantități tot mai mari de cereri simulate pe platformă pentru a verifica dacă poate sau nu să facă față a ceea ce este conceput să facă față.
- **Testarea stresului** face acest lucru cu un pas mai departe și este utilizată pentru a evalua modul în care software-ul dvs. va răspunde la sau peste sarcina sa maximă. Scopul testării stresului este de a supraîncărca aplicația în mod intenționat până când se rupe, aplicând atât scenarii de încărcare realiste, cât și nerealiste. Cu testarea la stres, veți putea găsi punctul de eșec al software-ului dvs.
- **Testarea de anduranță**, cunoscută și sub numele de testare prin înmuiere, este utilizată pentru a analiza comportamentul unei aplicații sub o anumită cantitate de sarcină simulată pe perioade mai lungi de timp. Scopul este să înțelegeți cum se va comporta sistemul dvs. în condiții de utilizare susținută, făcându-l un proces mai lung decât testarea sarcinii sau a stresului (care sunt concepute să se încheie după câteva ore). Un test critic de rezistență este că ajută la descoperirea scurgerilor de memorie.
- **Testarea vârfurilor** este un tip de testare a sarcinii utilizat pentru a determina modul în care software-ul dvs. va răspunde la explozii substanțial mai mari de activități simultane ale utilizatorului sau ale sistemului în perioade diferite de timp. În mod ideal, acest lucru vă va ajuta să înțelegeți ce se va întâmpla atunci când sarcina va crește brusc și drastic.

- Testarea securității

Odată cu creșterea platformelor de testare bazate pe cloud și a atacurilor cibernetice, există o preocupare și o nevoie crescândă pentru securitatea datelor utilizate și stocate în software. Testarea securității este o tehnică de testare software nefuncțională utilizată pentru a determina dacă informațiile și datele dintr-un sistem sunt protejate. Scopul este de a găsi în mod intenționat lacune și riscuri de securitate în sistem care ar putea duce la accesul neautorizat la sau pierderea informațiilor prin sondarea aplicației pentru puncte slabe. Există mai multe tipuri ale acestei metode de testare, fiecare dintre acestea având ca scop verificarea a șase principii de bază ale securității:

- Integritate
- Confidențialitate
- Autentificare
- Autorizare
- Disponibilitate
- Non-respingere

- Testarea utilizabilității

Testarea utilizabilității este o metodă de testare care măsoară ușurința utilizării unei aplicații din perspectiva utilizatorului final și este adesea efectuată în timpul etapelor de testare a sistemului sau de acceptare. Scopul este de a determina dacă designul vizibil și estetica unei

aplicații îndeplinesc sau nu fluxul de lucru prevăzut pentru diferite procese, cum ar fi conectarea la o aplicație. Testarea utilizabilității este o modalitate excelentă pentru echipe de a revizui funcții separate sau sistemul în ansamblu este intuitiv de utilizat.

- Testarea compatibilității

Testarea compatibilității este utilizată pentru a evalua modul în care o aplicație sau o bucată de software va funcționa în diferite medii. Este folosit pentru a verifica dacă platforma și componentele sale sunt compatibile cu mai multe sisteme de operare, platforme, browsere sau configurații de rezoluție. Scopul este de a vă asigura că funcționalitatea software-ului dvs. este acceptată în mod constant în orice mediu pe care vă așteptați să îl utilizeze utilizatorii finali.

- Rezultatele testării
  - Testarea unității

Metoda standard de testare a unității, preferată în cadrul Holisun pentru orice proiect Java, este folosirea jUnit Framework. jUnit este un cadru open-source încorporat în majoritatea cadrelor Java și IDE-urilor (cum ar fi NetBeans). Pentru fiecare clasă a cadrului, există un test definit, cu scopul de a acoperi 100% cod și, dacă nu este posibil, cât mai aproape de 100%. Fiecare ramură este verificată cu atenție, codul este refractat astfel încât metodele de clasă să fie cât mai atomice posibil.

Testele unitare se execută manual la momentul proiectării, pentru a se asigura că atât testele, cât și codul de lucru se comportă conform așteptărilor. Sunt scrise mai multe teste pentru fiecare metodă, verificând cele mai multe valori și combinații posibile pentru parametrii metodei. După efectuarea testelor și dacă toate trec, clasa testată, împreună cu metodele sale, ar trebui să accepte doar tipul / tipurile variabile corecte și numai intervalul valoric corect pentru fiecare dintre ele. Sistemul ar trebui să accepte erori și excepții și ar trebui să trateze oricare dintre ele cu grație și corect.

De asemenea, trebuie înțeles că utilizatorii pot introduce date nevalide (fie din greșeală, fie ca parte a unui atac). În ceea ce privește această situație, toate datele ar trebui să fie validate și evadate corect de fiecare modul, chiar dacă utilizatorul nu introduce date direct în acel modul (lacunele pot apărea în orice modul, deci sistemul ar trebui să aibă mecanisme de siguranță pentru această situație). Acest lucru este acoperit și de testarea unitară.

  - Testarea integrării

După finalizarea testării unitare și întregul sistem trece toate testele unitare, este timpul să testați sistemul (sau platforma în cazul nostru) ca întreg. Diferitele părți ale sistemului sunt testate în funcție de scenariile lor de utilizare (API vs acces Browser vs. acces local). O suită de valori este definită pentru toți parametrii necesari și toate funcțiile platformei sunt direcționate prin intermediul software-ului. Pentru această testare am folosit aplicația Open-Source Jenkins.

Jenkins este o platformă software de integrare continuă / implementare continuă (CI / CD), utilizată pentru automatizarea construcției corecte a întregii platforme software și, de asemenea, pentru a rula diferite cazuri de test pe software-ul construit cu succes, cazuri de testare care au fost definite anterior de echipa de testare . Jenkins este, de asemenea, capabil să publice rezultatele pe un server live (un server de stocare). Rezultatele testării sunt raportate înapoi echipei de dezvoltare, astfel încât problemele să poată fi rezolvate.

Similar cu jUnit, Jenkins folosește cazuri de testare definite de echipa de testare. Dar, spre deosebire de jUnit, Jenkins va fi aplicat pe întreaga platformă, concentrându-se pe colaborarea corectă a diferitelor module, schimbul corect de informații și, în cele din urmă, ieșirea corectă pentru diferite intrări.

Setarea implicită pentru Jenkins implică faptul că fluxul de lucru este declanșat de fiecare dată când există modificări ale codului. Jenkins scanează folderul GIT pentru noi confirmări, trasând modificările, construind și testând toate componentele cadrului, trecând astfel întregul cod prin conducta sa.

- Testarea sistemului

Odată ce întregul sistem trece toate testele și integrarea nu produce erori, echipa de testare trece prin unele teste manuale sau semi-automate ale sistemului. Diferite platforme software bazate pe Selenium sunt utilizate dacă partea testată a platformei are o componentă de afișare care necesită interacțiune cu mouse-ul sau tastatura.

Este definit un grafic de sistem și GraphWalker este utilizat, împreună cu cazurile de testare Selenium, pentru a simula toate rutele posibile pe care le-ar putea parcurge un utilizator atunci când folosește GUI, împreună cu stările aplicației. Testarea sistemului va scoate la iveală potențiale probleme ascunse, cum ar fi link-uri moarte, rutare incorectă sau probleme de interfață.

API-urile au fost testate cu o combinație de cod personalizat și aplicația gratuită Postman. Postman a permis echipei de testare să testeze și să depaneze manual funcțiile API. Aceste teste au fost limitate la etapele inițiale de dezvoltare, unde testarea rapidă este importantă, pentru a vedea dacă o funcție se comportă corect. Pentru teste mai aprofundate, jUnit a fost utilizat cu teste care au acoperit o gamă largă de valori posibile, precum și valori greșite, pentru a evalua corectitudinea acestor funcții.

Tabel 9 Verificare de securitate pentru asistență AR

➔ Toate componentele aplicației sunt identificate și cunoscute a fi necesare.	Da
➔ Facilitățile de stocare a credențialelor de sistem sunt utilizate în mod corespunzător pentru a stoca date sensibile, cum ar fi credențiale de utilizator sau chei criptografice.	Da
➔ Nu sunt scrise date sensibile în jurnalele de aplicații.	Da
➔ Nu sunt partajate date sensibile cu terți, cu excepția cazului în care este o parte necesară a arhitecturii.	Da
➔ Clipboard-ul este dezactivat pe câmpurile de text care pot conține date sensibile.	Da
➔ Nu sunt expuse date sensibile prin intermediul mecanismelor IPC.	Da

➔ Nu sunt expuse date sensibile, cum ar fi parolele sau pinii, prin interfața cu utilizatorul.	Da
➔ Nu sunt incluse date sensibile în copiile de rezervă generate de sistemul de operare mobil.	Da
➔ Aplicația elimină datele sensibile din vizualizări atunci când este în fundal.	Da
➔ Aplicația nu conține date sensibile în memorie mai mult decât este necesar, iar memoria este ștearsă explicit după utilizare.	Da
➔ Aplicația nu se bazează pe criptografie simetrică cu chei codificate ca o singură metodă de criptare.	Da
➔ Aplicația folosește implementări dovedite ale primitivelor criptografice.	Da
➔ Aplicația nu folosește protocoale criptografice sau algoritmi care sunt considerați pe scară largă depreciați din motive de securitate.	Da
➔ Toate valorile aleatoare sunt generate utilizând un generator de numere aleatorii suficient de sigur.	Da
➔ Datele sunt criptate în rețea folosind TLS (sau echivalente). Canalul securizat este utilizat în mod constant în toată aplicația.	Da
➔ Aplicația verifică certificatul X.509 al punctului final la distanță atunci când este stabilit canalul securizat. Sunt acceptate numai certificatele semnate de un CA de încredere.	Da
➔ Aplicația fie folosește propriul magazin de certificate, fie fixează certificatul de punct final sau cheia publică și ulterior nu stabilește conexiuni cu puncte finale care oferă un certificat sau o cheie diferită, chiar dacă este semnat de o autoritate de încredere	Da
➔ Aplicația solicită doar setul minim de permisiuni necesare.	Da

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

➔ WebViews sunt configurate pentru a permite doar setul minim de manipulare de protocol necesare (în mod ideal, doar https este acceptat). Handlerele potențial periculoase, cum ar fi fișierul, telul și aplicația-id, sunt dezactivate.	Vizualizările web nu sunt utilizate
➔ Codul de depanare a fost eliminat, iar aplicația nu înregistrează erori detaliate sau mesaje de depanare.	Da
➔ Toate componentele de terță parte utilizate de aplicația mobilă, cum ar fi bibliotecile și cadrele, sunt identificate și verificate pentru vulnerabilități cunoscute.	Da
➔ Aplicația prinde și gestionează posibile excepții.	Da
➔ Logica de gestionare a erorilor în controalele de securitate refuză accesul în mod implicit	Da
➔ Funcțiile de securitate gratuite oferite de lanțul de instrumente, cum ar fi minimizarea codurilor de octeți, protecția stivei, suportul PIE și numărarea automată a referințelor, sunt activate.	Da
➔ Aplicația previne depanarea și / sau detectează și răspunde la atașarea unui depanator. Toate protocoalele de depanare disponibile trebuie acoperite.	Da
➔ Toate fișierele și bibliotecile executabile care aparțin aplicației sunt fie criptate la nivel de fișier și / sau segmente importante de cod și date din interiorul executabilelor sunt criptate sau împachetate. Analiza statică banală nu dezvăluie codul sau date importante.	Da

Tabel 10 Verificări de securitate a bazei de date MySQL

➔ Securizați serverul care găzduiește instanța bazei de date MySQL	◆ Multe atacuri cunoscute sunt posibile numai după ce a fost dobândit accesul fizic la o mașină. Din acest motiv, cel mai bine este să aveți serverul de aplicații și serverul de baze de date pe diferite mașini. Dacă acest lucru nu este posibil, trebuie să vă asigurați că executați comenzi la distanță prin intermediul unui server de aplicații, în caz contrar, un atacator poate să vă dăuneze baza de date chiar și fără permisiuni. Din acest motiv, oricărui serviciu care rulează pe aceeași mașină ca baza de date ar trebui să i se acorde cele mai mici privilegii de permisiune care să permită serviciului să funcționeze fără probleme.	Da (pe diferite mașini virtuale)

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

➔ Dezactivați sau restricționați accesul la distanță	◆ Dacă se utilizează accesul la distanță, asigurați-vă că numai gazdele definite pot accesa serverul. Luați în considerare restricționarea MySQL de la deschiderea unui socket de rețea (inițierea unei conexiuni la distanță)	Da
➔ Dezactivați utilizarea LOCAL INFILE	◆ Dezactivați utilizarea comenzii „LOAD DATA LOCAL INFILE”, care va ajuta la prevenirea citirii neautorizate din fișierele locale.	Da
➔ Schimbați numele de utilizator și parola de root	◆ Numele de utilizator implicit al administratorului de pe serverul MySQL este „root”. Hackerii încearcă adesea să obțină acces la permisiunile sale. Pentru a face această sarcină mai dificilă, redenumiți „root” cu altceva și furnizați-i o parolă alfanumerică lungă și complexă.	Da
➔ Eliminați baza de date „test”	◆ MySQL vine cu o bază de date „test” destinată spațiului de testare. Acesta poate fi accesat de utilizatorul anonim și, prin urmare, folosită de numeroase atacuri.	Da
➔ Eliminați conturile anonime și învechite	◆ Baza de date MySQL vine cu unii utilizatori anonimi cu parole goale. Drept urmare, oricine se poate conecta la baza de date	Da
➔ Privilegiile reduse ale sistemului; creșteți securitatea bazei de date cu ajutorul controlului bazat pe roluri	◆ O recomandare foarte comună de securitate a bazei de date este reducerea permisiunilor acordate diferitelor părți. MySQL nu este diferit. De obicei, atunci când dezvoltatorii lucrează, aceștia folosesc permisiunea maximă a sistemului și acordă o atenție mai mică principiilor permisiunii decât ne-am putea aștepta. Această practică poate expune baza de date la riscuri semnificative	Da

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

➔ Privilegiile mai mici ale bazei de date	◆ Doar conturilor de administrator trebuie să li se acorde privilegiile SUPER / PROCESS / FILE și accesul la baza de date MySQL. De obicei, este o idee bună să reduceți permisiunile administratorului pentru accesarea datelor. Examinați privilegiile celorlalți utilizatori și asigurați-vă că acestea sunt setate corespunzător.	Da
➔ Schimbați directorul root	◆ Un chroot pe sistemele de operare Unix este o operație care schimbă directorul aparent al rădăcinii discului pentru procesul curent de rulare și pentru copiii acestuia. Un program reînrădăcinat într-un alt director nu poate accesa sau denumi fișierele din afara acestuia director, iar directorul este numit „închisoare chroot" sau (mai rar) „închisoare chroot". Prin utilizarea mediului chroot, accesul la scriere al proceselor MYSQL (și al proceselor copil) poate fi limitat, sporind securitatea serverului.	Da
➔ Eliminați istoricul	◆ În timpul procedurilor de instalare, există o mulțime de informații sensibile care pot ajuta un intrus să atace o bază de date.	Da

12. Concluzii Raport tehnic

Instrumentul de vizualizare a datelor face parte din sistemul cel mare de colectare și analiză a datelor ce țin de proiectul Big-Smart-Log, pe baza cazurilor de utilizare furnizate de Ekol. Instrumentul vizualizează datele referitoare la locațiile și picioarele diferitelor rute parcurse de mijloacele de transport de la compania Ekol.
Locațiile pot fi depozite, terminale de aeroport, terminale de autobuz, hub-uri de marfă, stații etc.

Sistemul și arhitectura acestuia vor fi integrate, respectiv verificate și validate în sarcina *T5.1. Dezvoltarea cadrului de vizualizare a datelor*, constatarea și rezultatele vor fi raportate în *D5.1. Cadrul de vizualizare și monitorizare.*

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

13. Artefacte

Artefactele obținute se împart în două categorii mari:

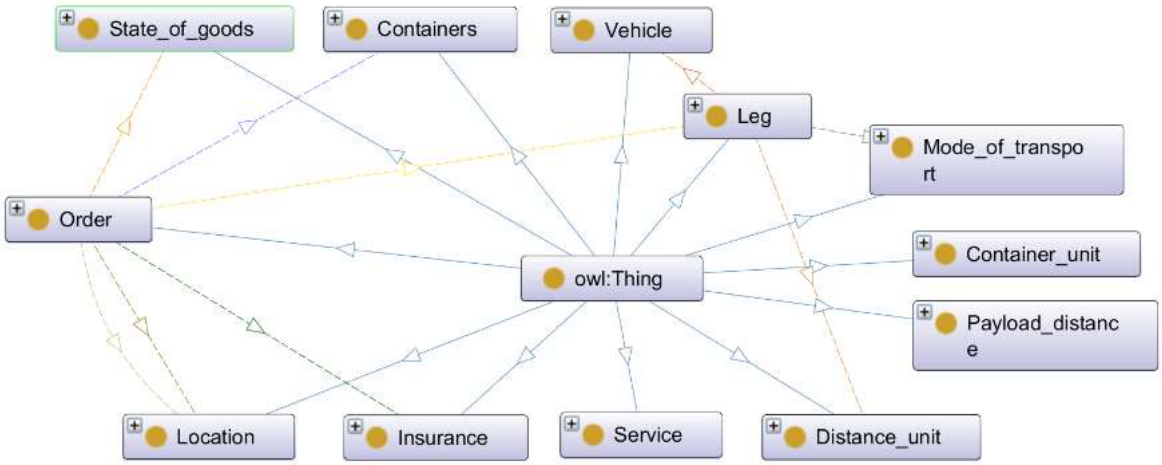
1. produse informatice
2. servicii informatice

Produsele informatice dezvoltate de HOLISUN în cadrul proiectului BIG-SMART-LOG sunt sumarizate în Table 1.

Table 1. Lista artefactelor elaborate în 2021

Denumire produs	Tip	Descriere	Dată
Platformă de optimizare a transportului multimodal	Produs Informatic	Platforma permite optimizarea rutelor, vehiculelor și, în general, a operațiunilor unei companii de transport multimodal.	mai 2021
Algoritmi de învățare automată pentru optimizarea transportului multimodal	Tehnologie	O serie de algoritmi utilizati în optimizarea transportului multimodal. Acești algoritmi stau la baza platformei de optimizare a transportului.	mai 2021
Ontologie pentru conceptualizarea transportului multimodal	Produs Informatic	O ontologie utilizată pentru definirea tuturor conceptelor și relațiilor referitoare la transportul (multimodal) într-o companie de profil. Ontologia este necesară în cadrul algoritmilor de optimizare.	ianuarie 2021

Conceptele principale împreună cu relațiile dintre ele cu care s-a lucrat în cadrul proiectului și pe baza cărora a fost creată ontologia sunt reprezentate în figura @@@.



Totodată, HOLISUN a lansat un nou serviciu pe piață, bazat pe rezultatele și know-how-ul obținut în cadrul proiectului:

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Denumire serviciu	Tip	Descriere	Dată
Servicii de optimizare a transportului multimodal	Serviciu Informatic	Pe lângă platforma de optimizare a transportului, HOLISUN furnizează și servicii de optimizare, fie sub forma de outsourcing, fie ca SaaS (software as a service).	mai 2021

14. Livrabile

Pe parcursul anului 2021, în cadrul proiectului au fost elaborate următoarele livrabile:

Nr. Livrabil	Termen	Denumire livrabil	Status
D5.2	M24	Report on the integration tests of the visualization and monitoring framework	Livrat 2021

15. Articole publicate

Articolele publicate în cadrul proiectului BIG-SMART-LOG sunt enumerate în Table 2.

Table 2. Articolele publicate în 2021

No	DOI	Type of publication	Link to the publication	Title	Authors	Publication	Relevant Pages	ISBN	Publisher
4		journal		Selective Survey of Saving Resources through Integrative Multimodal Transport	Oliviu Matei, Camelia Pintea, Rudolf Erdei	Informatica (Q1)			VUP
5		conference		Multimodal Transportation Overview andOptimization Ontology for a Greener	Oliviu Matei, Rudolf Erdei, Daniela Delinschi	10th Computer Science On-line Conference 2021			OpenPublish

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

16. Alte activități de diseminare

Table 3. Lista activităților de diseminare

Nr.	Eveniment	Locație / Organizator	Data	Participanți	URL	Rezultate
9	International Machines Forum	Zoom / b2match	11th and 12th of March 2021	Rudolf Erdei, Daniela Delinschi	https://machinery2021.b2match.io/	Prezentarea scenariilor de utilizare și a progreselor domeniului, în contextul lot și al optimizărilor
10	Cluster 3 HE	b2match	5th and 6th of May 2021	Rudolf Erdei, Daniela Delinschi	https://cluster3he.b2match.io	Prezentarea proiectului BSL și stabilirea unor posibile colaborări
11	B2B Software Days	b2match	10th to 12th of May 2021	Rudolf Erdei	https://2021.b2bsoftwaredays.com/	Prezentarea proiectului BSL și stabilirea unor posibile colaborări
12	DIGITAL ENTERPRISE SHOW 2021	b2match	18th to 19th of May 2021	Rudolf Erdei	https://des2021.b2match.io/	Prezentarea proiectului BSL și stabilirea unor posibile colaborări
13	#GIS2021 - Global Innovation Summit 2021	b2match	18th to 20th of	Daniela Delinschi	https://gis2021.b2match.io/	Prezentarea proiectului BSL și

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

			May 2021			stabilirea unor posibile colaborări
14	Green Opportunities with the EEA and Norway Grants	b2match	19th to 20th of May 2021	Oliviu Matei, Rudolf Erdei	https://green-opportunities-with-eea.b2match.io/	Prezentare a proiectului BSL și stabilirea unor posibile colaborări
15	ITmatch – virtual IT/ICT cooperation day 2021	b2match	25th of May 2021	Rudolf Erdei, Daniela Delinschi	https://itmatch-virtual-it-ict-cooperation.b2match.io/	Prezentare a proiectului BSL și stabilirea unor posibile colaborări

Au fost desfășurate o serie de activități de diseminare în cadrul unor evenimente de afaceri, expoziții și evenimente de brokeraj sau networking, listate în Table 3.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Referințe

(Cmiil, 2014) Ćmil, Michał, Michał Matłoka, and Francesco Marchioni. Java EE 7 development with WildFly. Packt Publishing Ltd, 2014.

(Richards, 2015) Richards, Mark. Software architecture patterns. Vol. 4. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Incorporated, 2015.

(Vassiliadis, 2009) Vassiliadis, Panos. "A survey of extract–transform–load technology." International Journal of Data Warehousing and Mining (IJDWM) 5, no. 3 (2009): 1-27.

(Bansal, 2014) Bansal, Srividya K. "Towards a semantic extract-transform-load (ETL) framework for big data integration." In 2014 IEEE International Congress on Big Data, pp. 522-529. IEEE, 2014.

(Henry, 2005) Henry, Scott, Sherlynn Hoon, Meeky Hwang, Diane Lee, and Michael D. DeVore. "Engineering trade study: extract, transform, load tools for data migration." In 2005 IEEE Design Symposium, Systems and Information Engineering, pp. 1-8. IEEE, 2005.

(Caserta, 2013) Caserta, Joe, and Ralph Kimball. The Data Warehouseetl Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley, 2013.

(Gunning, 2017) Gunning, David. "Explainable artificial intelligence (xai)." Defense Advanced Research Projects Agency (DARPA), and Web 2, no. 2 (2017).

(Miller, 2010) Miller, Frederic P., Agnes F. Vandome, and John McBrewster. Apache Maven. Alpha Press, 2010.

(Sauri, 2012) Saurí, Sergi, Pau Morales-Fusco, Maria Toledano, and Enrique Martín. "Empirical Analysis of Resiliency of Terminal Operations for Roll-On–Roll-Off Vessels." Transportation research record 2273, no. 1 (2012): 96-105.

(Gorton, 2011) Gorton, Ian. "Software quality attributes." Essential Software Architecture. Springer, Berlin, Heidelberg, (2011). 23-38.

(Haklay, 2008) Haklay, Mordechai, and Patrick Weber. "Openstreetmap: User-generated street maps." IEEE Pervasive Computing 7, no. 4 (2008): 12-18.

(Karich, 2014) Karich, Peter, and Stefan Schröder. "Graphhopper." http://www. graphhopper. com, last accessed 4, no. 2 (2014): 15.

(McHugh, 2013) McHugh, Bibiana. "Pioneering open data standards: The GTFS Story." Beyond transparency: open data and the future of civic innovation (2013): 125-135.

(Chen, 2017) Chen, Yiqun, Abbas Rajabifard, and Jennifer Day. "An advanced web API for isochrones calculation using OpenStreetMap data." In International Conference on Computers in Urban Planning and Urban Management, pp. 185-205. Springer, Cham, 2017.

(Noto, 2000) Noto, Masato, and Hiroaki Sato. "A method for the shortest path search by extended Dijkstra algorithm." In *Smc 2000 conference proceedings. 2000 ieee international*

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0, vol. 3, pp. 2316-2320. IEEE, 2000.

(Nosrati, 2012) Nosrati, Masoud, Ronak Karimi, and Hojat Allah Hasanvand. "Investigation of the*(star) search algorithms: Characteristics, methods and approaches." *World Applied Programming* 2, no. 4 (2012): 251-256.

(Geisberger, 2008) Geisberger, Robert, Peter Sanders, Dominik Schultes, and Daniel Delling. "Contraction hierarchies: Faster and simpler hierarchical routing in road networks." In *International Workshop on Experimental and Efficient Algorithms*, pp. 319-333. Springer, Berlin, Heidelberg, 2008.

(Steadie Seifi, et al, 2014) M. SteadieSeifi et al., “Multimodal freight transportation planning: A literature review,” European Journal of Operational Research, vol. 233, no. 1, pp. 1–15, 2014.

(Steneker2016)Steneker, Maikel. Towards an empirical validation of the TIOBE Quality Indicator. Diss. Eindhoven University of Technology, 2016.

(W.-J. Van Schijndel, 2000) and J. Dinwoodie, “Congestion and multimodal transport: a survey of cargo transport operators in the Netherlands,” Transport Policy, vol. 7, no. 4, pp. 231–241, 2000.

(T. Litman, 2017), Introduction to multi-modal transportation planning. Victoria Transport Policy Institute Canada, 2017.

(M. Zamanifar, 2020) and T. Hartmann, 2020, “Literature review of optimization based decision model for disaster recovery planning of transportation network,” DepositOnce TU Berlin, 2020, 8 pages.

(Z. Peplowska-Dabrowska, 2019) and J. Nawrot, Codification of Maritime Law:

Challenges, Possibilities and Experience. (Taylor & Francis, 2019), 2019.

(K. G. Zografos, 2008) and K. N. Androutsopoulos, “Algorithms for itinerary planning in multimodal transportation networks,” IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 1, pp. 175–184, 2008.

(Y. Y. G. Jianya, 1999), “An efficient implementation of shortest path algorithm based on Dijkstra algorithm,” Journal of Wuhan Technical University of Surveying and Mapping, vol. 3, no. 004, 1999.

(B. L. Golden, 1977), T. L. Magnanti, and H. Q. Nguyen, “Implementing vehicle routing algorithms,” Networks, vol. 7, no. 2, pp. 113–148, 1977.

(M. Zhang, 2013), B. Wiegmans, and L. Tavasszy, “Optimization of multimodal networks including environmental costs: a model and findings for transport policy,” Computers in industry, vol. 64, no. 2, pp. 136–145, 2013.

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com

Telefon: +40262226778
WEB: http://www.holisun.com • E-mail: info@holisun.com