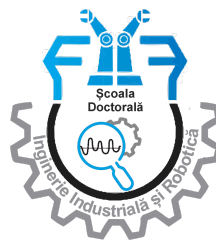




MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
Universitatea Națională de Știință și Tehnologie
POLITEHNICA București

Școala Doctorală de
Inginerie Industrială și Robotică



Andra-Paula AVASILOAIE

TEZA DE DOCTORAT

**Contribuții la dezvoltarea abordărilor
bazate pe inteligență artificială pentru
descoperirea cerințelor software și
generarea de user stories**

Coordonator științific,
prof.univ.habil.dr.ing.ec.mat. Augustin SEMENESCU
(POLITEHNICA Bucharest)

- 2025 -

Cuprins

Introduction	3
Chapter 1. Research Framework	5
Chapter 2. Software Development Ecosystem and Methodologies	6
2.1 Current Software Development Ecosystem	6
2.2 Project Management in Software Development	6
2.3 Software Development Life Cycle (SDLC) Models and Their Evolution	6
2.4 Agile vs. Waterfall: Comparative Perspectives	7
2.5 Agile Principles and Values	8
2.6 Challenges in Managing Software Development Projects	9
Chapter 3. Roles and Responsibilities in Agile Teams	10
3.1 The Role of the Business Analyst	10
3.2 The Role of the Product Owner and the Product Manager	10
3.3 Comparative Scenarios of Role Distribution	10
3.4 Research Findings	11
Chapter 4. Requirements Engineering and User Stories	12
4.1 Sources and Analysis of Requirements	12
4.2 The Concept and Structure of User Stories	13
4.3 Epics, Acceptance Criteria, and Definition of Ready	13
4.4 Limitations of Traditional User Story Writing	13
Chapter 5. Standardization of User Story Writing	15
5.1 Proposed New Standard for User Story Definition	15
5.2 Validation through Practice and Training	15
5.3 Towards Tool-Supported Standardization	16
5.4 From Templates to Automated Generation	16
Chapter 6. The Jira Extension for Deterministic Automation of User Story Writing	17
6.1 Jira as a Platform for Agile Requirements Management	17
6.2 Motivation for Developing the Jira Extension	17
6.3 Design and Functionality of Specs-Assistant	17
6.4 User Interface of the Extension	18
6.5 Benefits Observed in Practice	19

Chapter 7 – AI-powered User Story Generation Interface	20
7.1 Related Work	20
7.2 From Manual Templates to AI-Enhanced Generation	20
7.3 Comparative Evaluation of AI Models	23
7.4 Findings and Challenges	24
Chapter 8. AI-Driven Multimodal Requirement Discovery	25
8.1 Foundational Technologies in AI-Driven Requirement Discovery	25
8.2 From Fragmented Tools to Integrated Discovery Pipelines	26
8.3 Dual-Phase Generation Approach	26
8.4 System Overview	26
8.5 Inference Workflow	29
Chapter 9: A Simplified Iteration of Discovery AI with GPT-5	31
9.1 System Design	31
9.2 Comparative Analysis: Old vs. New Iteration	32
9.3 Interface and Usability	33
9.4 Opportunities and Limitations	35
Chapter 10. Conclusions	36
10.1. Concluzii generale	36
10.2. Original Contributions	36
10.3. Future Research Directions	37
10.4. Work Synthesis	37
References	38

Introducere

Cercetarea doctorală desfășurată în această teză reflectă atât motivația academică, cât și experiența profesională a autoarei, cu accent pe avansarea domeniului ingineriei cerințelor software prin abordări deterministe și bazate pe inteligență artificială. Pornind de la provocările întâlnite ca Business Analyst, lucrarea subliniază importanța critică a reducerii decalajelor de comunicare dintre stakeholderi și echipele de dezvoltare și a traducerii inputurilor de business ambigue în specificații precise și acționabile.

Studiul este motivat de complexitatea în creștere a proiectelor software în contextul transformării digitale și al adoptării tot mai largi a metodologiilor Agile. Deși Agile a introdus flexibilitate și colaborare în dezvoltarea software, descoperirea cerințelor rămâne o provocare constantă. User stories, deși utilizate pe scară largă ca artefacte ușoare pentru captarea nevoilor clienților, suferă adesea de ambiguitate, inconsistență și efort suplimentar de formare. Experiențele profesionale timpurii ale autoarei, inclusiv workshopuri de training livrate către peste șaptezeci de analiști și product owners în formare (dintre care peste 80% au fost ulterior angajați în domeniu), au oferit dovezi empirice ale acestor probleme și au modelat direcția cercetării.

Introducerea identifică două blocaje majore:

- distribuția inegală a responsabilităților în echipele Agile, în funcție de prezența sau absența unor roluri precum Business Analyst, Product Owner sau Project Manager;
- timpul semnificativ necesar pentru a forma, structura și standardiza user stories, în ciuda aparentei lor simplități.

Cercetările inițiale au explorat aceste provocări prin analiza distribuției rolurilor în echipe Agile și prin propunerea unui nou standard pentru redactarea user stories. Experimentele timpurii au evidențiat îmbunătățiri măsurabile în claritate și eficiență, dar au subliniat și dificultatea inerentă de a extrage cerințe direct din canale de comunicare nestructurate, precum interviuri cu stakeholderi, workshopuri și întâlniri de business.

În paralel, progresele în inteligența artificială, în special modelele lingvistice de mari dimensiuni (LLMs), au introdus noi posibilități pentru transformarea ingineriei cerințelor. Recunoașterea automată a vorbirii (ASR), generarea augmentată prin căutare (RAG) și generarea de date sintetice permit acum sisteme multimodale care procesează date brute din întâlniri de business și produc artefacte Agile structurate. Totuși, majoritatea lucrărilor anterioare tratează cerințele ca

artefacte exclusiv textuale, ignorând dimensiunile multimodale ale comunicării (intonare, accent, context comun), care influențează puternic semnificația [1][2].

Teza este structurată în zece capitole, fiecare reprezentând o etapă a traiectoriei de cercetare:

Capitolul 1 prezintă motivația, obiectivele, domeniul și metodologia cercetării.

Capitolul 2 oferă o revizuire extinsă a literaturii despre ingineria cerințelor, structura user stories, utilizarea AI în dezvoltarea software și prompt engineering pentru LLMs.

Capitolul 3 descrie metodologia cercetării, cu accent pe procesul iterativ de design și pe abordarea mixtă ce combină design exploratoriu, dezvoltarea de prototipuri și feedback-ul utilizatorilor.

Capitolul 4 prezintă o serie de studii de caz unde formatele deterministe și standardele timpurii pentru user stories au fost testate în contexte educaționale și profesionale.

Capitolul 5 introduce standardul Extended User Story (EUS), un nou model de redactare a user stories care îmbunătățește claritatea, trasabilitatea și alinierea prin reguli vizuale și convenții structurale.

Capitolul 6 descrie dezvoltarea și implementarea unei extensii JIRA care operationalizează standardul EUS prin template-uri reutilizabile și elemente de interfață dedicate Business Analysts și Product Owners.

Capitolul 7 avansează procesul de automatizare printr-o interfață AI care generează user stories din prompturi text folosind API-ul OpenAI GPT.

Capitolul 8 prezintă cea mai complexă soluție — un sistem multimodal de descoperire a cerințelor bazat pe AI.

Capitolul 9 explorează o versiune simplificată a arhitecturii Discovery AI, bazată pe capabilitățile multimodale ale GPT-5.

Capitolul 10 încheie teza printr-un rezumat al concluziilor, contribuțiilor și direcțiilor viitoare de cercetare.

Această evoluție progresivă ilustrează tranziția de la abordări deterministe la arhitecturi avansate bazate pe AI, demonstrând atât contribuții academice, cât și aplicabilitate practică în industrie.

Capitolul 1. Cadrul de cercetare

Ingineria cerințelor reprezintă o etapă critică în ciclul de viață al dezvoltării software (SDLC), însă adesea suferă de ambiguitate, refaceri costisitoare și depășiri de proiect [1], iar managementul deficitar al cerințelor este identificat ca o cauză principală a eșecurilor proiectelor software [2]. Metodologiile Agile se bazează pe user stories [3], dar asigurarea clarității, consistenței și a unor criterii de acceptare bine formulate rămâne o provocare constantă [4]. Instrumentele deterministe precum extensiile Jira au îmbunătățit eficiența, economisind până la 60% din efort [5], însă rămân dependente de domeniu și nu reușesc să surprindă cerințele ascunse în inputuri nestructurate, cum sunt interviurile și întâlnirile [6]. Deși AI și NLP au demonstrat potențial [7], majoritatea soluțiilor reduc în continuare cerințele la artefacte exclusiv textuale, neglijând natura multimodală a comunicării [8].

Această teză abordează aceste limitări prin cinci întrebări de cercetare, concentrându-se pe configurațiile de roluri în echipă (RQ1), standardizarea redactării user stories (RQ2), evaluarea instrumentelor deterministe (RQ3), proiectarea unor abordări multimodale bazate pe AI ce integrează ASR, RAG și LLMs (RQ4) și evaluarea comparativă a sistemelor deterministe față de cele bazate pe AI (RQ5). Obiectivele sunt aliniate acestor întrebări, acoperind de la analiza responsabilităților în echipă până la proiectarea și evaluarea atât a soluțiilor deterministe, cât și a celor bazate pe AI.

Contribuțiile cercetării se extind pe mai multe planuri: teoretice (un cadru care leagă rolurile Agile de cerințe și un standard structurat pentru user stories [4]), metodologice (chestionare, workshopuri și studii de caz), practice (o extensie Jira care demonstrează câștiguri de eficiență măsurabile [5]), tehnologice (un sistem multimodal bazat pe AI care integrează ASR, RAG și LLMs [5]) și empirice (evaluări bazate pe claritate, trasabilitate și scalabilitate).

Metodologic, lucrarea utilizează o abordare mixtă—metode calitative (workshopuri, chestionare, interviuri), indicatori cantitativi (timp, claritate, eficiență), dezvoltarea de prototipuri și evaluarea comparativă a abordărilor manuale, deterministe și bazate pe AI.

Prin îmbinarea cercetării academice cu practica industrială, teza propune atât avansuri conceptuale, cât și operaționale pentru îmbunătățirea procesului de descoperire a cerințelor și a generării de user stories.

Capitolul 2. Ecosistemul și metodologiile de dezvoltare software

2.1 Ecosistemul dezvoltării software

Ecosistemul dezvoltării software s-a schimbat profund în ultimele decenii, evoluând dintr-o disciplină tehnică restrânsă într-un factor strategic al competitivității organizaționale. Odată cu globalizarea, echipele distribuite și presiunile de reglementare, proiectele software de astăzi trebuie să integreze nu doar practici ingineresti, ci și strategie de afaceri, conformitate și experiența utilizatorilor. Mediul este adesea descris ca VUCA — volatil, incert, complex și ambiguu — ceea ce necesită metodologii adaptive [9]. Pandemia COVID-19 a accelerat această tendință, forțând companiile să treacă rapid către produse și servicii digitale [10].

Dezvoltarea software acoperă trei categorii majore: software de sistem, software de programare și software de aplicație. Aceste domenii sunt din ce în ce mai interconectate, iar proiectele moderne necesită integrarea perspectivelor tehnice și de business. Spre deosebire de trecut, când dezvoltarea software era în primul rând despre a scrie cod funcțional, acum ea face parte din strategiile de transformare digitală la nivel de întreprindere.

Managementul proiectelor asigură alinierea ideilor, oamenilor și resurselor pentru a livra valoare. Ciclul clasic — inițiere, planificare, execuție, monitorizare și închidere — rămâne relevant, dar secvența sa rigidă a fost contestată de realitățile mediilor volatile. Practicile Agile au introdus iterația, buclele de feedback și adaptarea continuă, ceea ce face proiectele mai reziliente. Totuși, persistă probleme recurente: derapaje de scop cauzate de schimbarea cerințelor, dificultatea de a echilibra viteza și calitatea în limitele bugetului și nevoia de a coordona eficient echipe distribuite. Noi categorii de riscuri, precum securitatea cibernetică și utilizarea etică a AI, complică suplimentar peisajul proiectelor [11].

2.2 Modelele ciclului de viață al dezvoltării software (SDLC) și evoluția lor

SDLC structurează dezvoltarea în faze de: cerințe, design, implementare, testare, implementare și mentenanță [12]. De-a lungul timpului, au fost propuse mai multe modele:

- **Modelul Waterfall** urmează o secvență liniară strictă, asigurând trasabilitate dar oferind puțină flexibilitate [1].
- **Modelul Spiral** (Boehm, 1986) integrează iterația cu management explicit al riscurilor, fiind potrivit pentru proiecte complexe și cu risc ridicat [14].

- **Modelul V** aliniază fiecare etapă de dezvoltare cu activități corespunzătoare de testare, consolidând validarea dar crescând rigiditatea [1].
- **Modelul Agile** abandonează linearitatea, concentrându-se pe cicluri iterative, feedback al stakeholderilor și livrare incrementală [15].

2.3 Agile vs. Waterfall: Perspective comparative

Modelul Waterfall, introdus de Royce în 1970, a stabilit o abordare secvențială a dezvoltării software. A fost larg adoptat în industrii precum aerospațială, apărare și guvern, unde conformitatea, predictibilitatea și documentația extinsă erau esențiale [16]. Totuși, rigiditatea lui a dus adesea la costuri mari atunci când cerințele evoluau pe parcursul unui proiect.

Prin contrast, Manifestul Agile (2001) a marcat o schimbare culturală și metodologică, promovând adaptabilitatea, colaborarea și livrarea incrementală [17]. Practicile Agile încurajează implicarea frecventă a stakeholderilor și cicluri iterative, ceea ce le face mai potrivite pentru piețe dinamice.

Diferențele dintre cele două paradigme sunt rezumate în Tabelul 2.1.

Tabelul 2.1. Compararea metodologiilor Waterfall și Agile

Criteriu	Model Waterfall	Metodologia Agile
Structura procesului	Faze liniare, secvențiale (cerințe → implementare).	Cicluri iterative, incrementale (sprinturi, livrare continuă).
Flexibilitate	Redusă – schimbările sunt costisitoare după închiderea unei faze.	Ridicată – cerințele pot evolua pe parcursul dezvoltării.
Implicarea clientului	Limitată – mai ales la început și la livrarea finală.	Continuă – implicare activă la fiecare iterație.
Documentație	Detaliată, specificații cuprinzătoare.	Minimalistă – accent pe software funcțional.
Managementul riscurilor	Riscurile sunt adesea descoperite târziu.	Riscurile sunt abordate incremental, în timpul iterațiilor.

Livrare	O singură lansare finală la sfârșitul proiectului.	Lansări frecvente de incrementuri funcționale.
Cel mai potrivit pentru	Cerințe stabile, proiecte orientate pe conformitate.	Medii dinamice cu cerințe în schimbare.

Abordările hibride sunt din ce în ce mai comune, combinând guvernanta Waterfall cu flexibilitatea Agile. Studiile din industrie confirmă că peste 70% dintre organizații folosesc practici hibride, mai ales în proiecte din finanțe, sănătate și sectorul guvernamental [18].

Pentru ilustrare vizuală, Figura 2.4 prezintă fluxul liniar al Waterfall comparat cu iterațiile ciclice ale Agile.

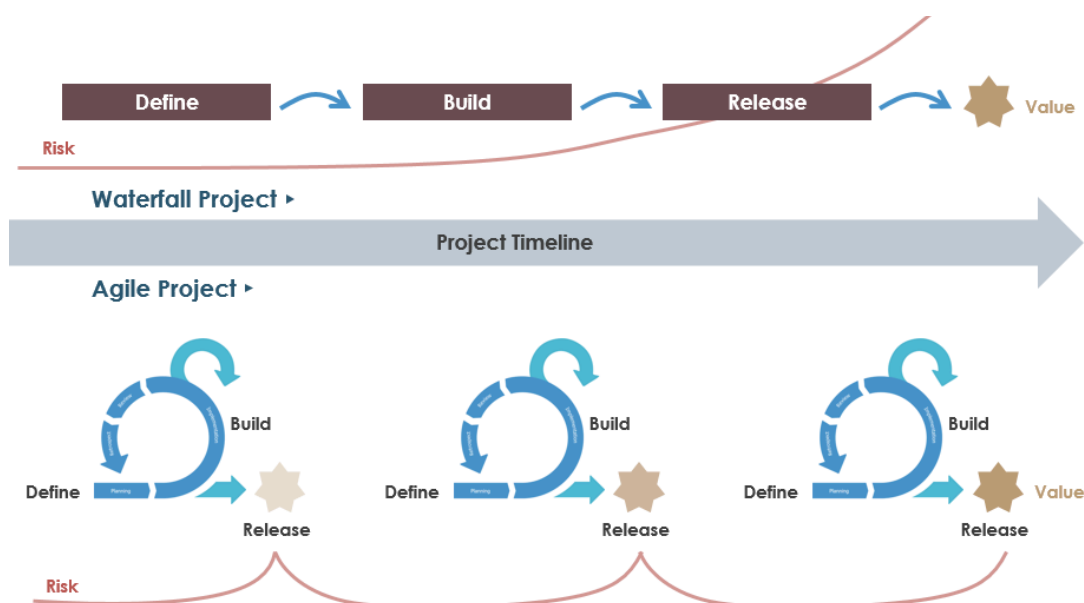


Figure 2.4. Waterfall vs. Modele Agile de Dezvoltare

*Sursa: "Agile Product Development vs. Waterfall: Choosing the Right Approach,"
<https://guides.visual-paradigm.com/>*

2.4 Principiile și valorile Agile

Manifestul Agile a enunțat patru valori fundamentale: indivizi și interacțiuni, software funcțional, colaborare cu clientul și răspuns la schimbare [17]. Acestea sunt întărite de douăsprezece principii ce promovează livrarea timpurie, ritm sustenabil, excelență tehnică, simplitate și auto-organizare. De-a lungul a două decenii, aceste valori s-au dovedit remarcabil

de durabile și s-au extins dincolo de echipele IT, influențând modele de leadership, strategii de inovare și cultura organizațională [21].

Agile este o familie de cadre cu implementări diferite.

- Scrum, cel mai răspândit, structurează munca în sprinturi cu roluri și ceremonii definite pentru a asigura livrarea și adaptarea.
- Kanban pune accent pe fluxul continuu prin vizualizarea sarcinilor și limitarea WIP, fiind preferat în contexte operaționale [23].
- SAFe, la scară mare, integrează Agile, Lean și DevOps, aliniind strategia cu execuția prin incrementuri de program, trenuri de livrare și planificare la nivel de portofoliu [24].

2.5 Provocări în managementul proiectelor software

Chiar și cu cadre mature, proiectele software întâmpină adesea dificultăți. Multe eșecuri nu se datorează metodologiei în sine, ci distribuției neclare a responsabilităților, alinierii slabe între echipele de business și cele tehnice și lipsei de asumare a riscurilor emergente. Echipele distribuite introduc bariere suplimentare de comunicare, iar mediile reglementate cer responsabilitate în domenii precum protecția datelor și guvernanta AI.

Aceste provocări persistente evidențiază necesitatea de a examina modul în care sunt structurate rolurile și responsabilitățile în echipele Agile — un subiect tratat în capitolul următor.

Capitolul 3. Roluri și responsabilități în echipele Agile

Metodologiile Agile pun accent pe echipe auto-organizate și multifuncționale, dar lasă flexibilitate în definirea modului în care sunt distribuite rolurile specifice. Această flexibilitate este atât un punct forte, cât și o provocare. Pe de o parte, permite organizațiilor să adapteze practicile Agile la propriul context; pe de altă parte, poate genera confuzie sau suprapuneri în responsabilități. Modul în care roluri precum Business Analyst (BA), Product Owner (PO) și Project Manager (PM) sunt alocate are un impact direct asupra modului în care cerințele sunt descoperite, documentate și prioritizate [23].

3.1 Rolul Business Analystului

Business Analystul este adesea legătura dintre stakeholderi și echipele de dezvoltare. În contexte Agile, contribuția sa depășește documentarea tradițională: facilitează rafinarea backlog-ului, asigură claritatea user story-urilor și definește criteriile de acceptare. În lipsa unui BA, echipele se confruntă adesea cu ambiguitate, ceea ce duce la cerințe neclare și criterii de acceptare inconsistente [23][24].

3.2 Rolul Product Ownerului și al Product Managerului

Product Ownerul este responsabil pentru maximizarea valorii produsului prin gestionarea backlog-ului și prioritizarea funcționalităților. El acționează ca vocea clientului, traducând nevoile de business în elemente de backlog. Product Managerul, în schimb, operează la un nivel strategic mai înalt, definind viziunea produsului, roadmap-ul și poziționarea pe termen lung [25]. În echipe mai mici, responsabilitățile PO și PM se pot suprapune, dar în cadrele Agile la scară largă, separarea lor este crucială pentru a menține claritatea.

3.3 Scenarii comparative de distribuire a rolurilor

Teza identifică și evaluează patru configurații tipice de roluri în echipele Agile. Aceste scenarii au fost testate prin workshopuri cu practicieni, iar rezultatele arată cum diferă claritatea și eficiența în funcție de componența echipei [28].

Tabel 3.1. Scenarii de distribuire a rolurilor în echipele Agile

Scenariu	Puncte forte	Puncte slabe
Doar BA	Specificații de calitate, user story-uri bine structurate.	Lipsă de prioritizare strategică; alinierea cu business-ul mai slabă.

Doar PO	Aliniere puternică la business, focus direct pe client.	Ambiguitate în cerințe, criterii de acceptare slabe.
BA + PO	Echilibru: claritate în user story-uri + prioritzare solidă.	Necesită colaborare; risc de suprapunere a rolurilor.
BA + PO + PM	Distribuție clară: BA asigură claritate, PO gestionează valoarea, PM garantează aria și riscurile.	Costuri mai mari de coordonare, potențiale conflicte între roluri.

Analiza a arătat că configurația BA + PO a oferit constant cea mai mare claritate și eficiență în ingineria cerințelor. Adăugarea unui PM a crescut coordonarea și guvernanta, dar uneori a introdus suprapuneri de roluri.

3.4 Rezultatele cercetării

Cercetarea confirmă că claritatea rolurilor este esențială pentru succesul Agile. Echipele fără BA s-au confruntat frecvent cu ambiguitate, iar echipele fără PO au dus lipsă de direcție puternică din partea business-ului. Coexistența BA și PO a produs cel mai bun echilibru.

În concluzie, proiectele Agile nu beneficiază de eliminarea rolurilor, ci de reinterpretarea și echilibrarea lor în funcție de contextul proiectului și nevoile organizaționale.

Capitolul 4. Ingineria cerințelor și user story-uri

Acest capitol examinează modul în care cerințele sunt descoperite, analizate și exprimate în medii Agile, cu un accent deosebit pe user story-uri ca artefact dominant. Capitolul evidențiază tranziția de la ingineria tradițională a cerințelor la practicile Agile, explorează structura și principiile user story-urilor și identifică principalele limitări care motivează dezvoltarea unui standard îmbunătățit.

4.1 Sursele și analiza cerințelor

Ingineria cerințelor este definită ca procesul sistematic de descoperire, documentare și gestionare a serviciilor și constrângerilor unui sistem. În abordările tradiționale, orientate pe planificare, cerințele erau stabilite din start ca repere contractuale. Metodologiile Agile contestă această viziune, tratând cerințele ca artefacte evolutive, co-create, dezvoltate iterativ prin colaborare.

Cerințele pot proveni din:

- stakeholderi (utilizatori, manageri, autorități de reglementare, sponsori) cu obiective divergente,
- procese de business, care conturează fluxurile funcționale,
- sisteme existente, care constrâng designul dar scot la iveală așteptări,
- piață și competitori, care stabilesc repere,
- reglementări (de ex. GDPR), care impun reguli obligatorii de conformitate.

Metodele cheie includ interviuri, workshopuri, analiza documentelor, observația, prototiparea și chestionarele. În Agile, aceste activități se repetă pe parcursul ciclului de viață, cu cerințe rafinate treptat în story-uri, epicuri și criterii de acceptare.

Agile introduce provocări precum: cerințe care apar continuu, dependența de input nestructurat, lipsa unei asumări clare a rolurilor (BA, PO, PM), dificultăți în alinierea stakeholderilor și trasabilitate mai slabă comparativ cu RE tradițională.

Schimbarea este atât culturală, cât și metodologică. Cerințele trec de la statutul de „contracte” statice la „conversații” continue. Analistii trebuie să echilibreze ambiguitatea cu claritatea, să documenteze „atât cât trebuie” și să mențină comunicarea continuă. Acest context pregătește terenul pentru user story-uri ca artefact preferat în Agile.

4.2 Conceptul și structura user story-urilor

User story-urile au apărut la sfârșitul anilor 1990 în Extreme Programming și au fost popularizate de Jeffries și Cohn ca alternative concise la documentele de cerințe. Structura lor standard este:

Ca [utilizator], vreau [obiectiv] pentru a [motivație].

Aceasta surprinde rolul, obiectivul și rațiunea.

Modelul INVEST al lui Bill Wake definește story-urile eficiente ca fiind: independente, negociabile, valoroase, estimabile, mici și testabile. În ciuda popularității, echipele se confruntă frecvent cu formulări vagi, criterii incomplete și story-uri prea mari.

Adaptări ale formatului includ Job Stories (axate pe context și motivație), Behavior-Driven Development (BDD) cu criterii de tip Given–When–Then și Story Mapping pentru a vizualiza parcursuri și dependențe.

Problemele comune sunt ambiguitatea, supra-simplificarea, rafinarea consumatoare de timp și adoptarea inconsistentă între echipe. Workshopurile și sondajele au confirmat că redactarea story-urilor consumă disproporționat de mult efort, mai ales când inputul provine din surse nestructurate.

4.3 Epicuri, criterii de acceptare și Definition of Ready

Cerințele Agile sunt organizate pe niveluri de abstractizare.

- Epicurile reprezintă funcționalități mari, împărțite în mai multe story-uri, dar adesea devin supradimensionate sau învechite.
- Criteriile de acceptare definesc condițiile de finalizare, fiind din ce în ce mai des formulate în sintaxă BDD, dar lipsesc frecvent sau sunt inconsistente.
- Definition of Ready (DoR) servește ca listă de verificare pentru elementele din backlog înainte de planificarea sprintului, dar este adesea aplicată inconsistent sub presiunea livrării.

Aceste practici, deși utile, suferă de ambiguitate, efort suplimentar și inconsistență, ceea ce evidențiază nevoia de standardizare mai clară.

4.4 Limitările scrierii tradiționale de user story-uri

User story-urile, deși utilizate pe scară largă, prezintă limitări recurente:

- ambiguitate datorată descrierilor în limbaj natural,
- inconsistență între echipe și organizații,
- neglijarea cerințelor non-funcționale precum performanța sau conformitatea,
- efort mare pentru a produce story-uri conforme INVEST,
- trasabilitate slabă, care necesită unelte externe,
- dificultăți în gestionarea interacțiunilor complexe, a constrângerilor tehnice sau a vizualurilor precum wireframe-uri.

Aceste limitări subliniază faptul că șablonul minimalist este insuficient pentru contexte la scară largă sau reglementate. Workshopurile și sondajele au confirmat investiția semnificativă de timp și variabilitatea calității în redactarea story-urilor.

Capitolul concluzionează că user story-urile trebuie să evolueze. Limitările identificate motivează dezvoltarea cadrului Extended User Story (EUS), care integrează criterii de acceptare structurate, descrieri detaliate și suport pentru elemente tehnice și vizuale. Acest standard va fi prezentat în capitolul următor.

Capitolul 5. Standardizarea scrierii user story-urilor

Acest capitol introduce cadrul Extended User Story (EUS), dezvoltat pentru a aborda limitările formatelor tradiționale de user story. El subliniază nevoia de standardizare în cerințele Agile, descrie regulile formatului EUS și prezintă validarea sa în practică. Capitolul urmărește, de asemenea, progresul de la șabloane la automatizare, culminând în sisteme capabile să transforme inputurile nestructurate de descoperire în elemente structurate de backlog.

5.1 Noul standard propus pentru definirea user story-urilor

Studiile confirmă că cerințele prost definite sunt cauza majorității eșecurilor proiectelor software. Experiența profesională ca Business Analyst și Product Owner, combinată cu rezultatele sondajelor, a arătat că șabloanele tradiționale de story produceau atât ambiguitate, cât și ineficiență. Aproximativ 20% din timpul necesar pentru redactarea unui user story era irosit pe sarcini de formatare.

Formatul Extended User Story (EUS) a fost dezvoltat pentru a reduce această pierdere și pentru a oferi o abordare standardizată, robustă. El introduce reguli structurale și vizuale:

- Delimitarea secțiunilor: overview (AS A / I WANT TO / SO THAT), descriere, criterii de acceptare în Gherkin.
- Convenții de formatare: cuvinte cheie scrise cu majuscule; butoane evidențiate cu violet; mesaje de eroare în roșu italic între ghilimele; mesaje informaționale în albastru italic între ghilimele.
- Descrieri structurate: tabele, liste și referințe la story-uri conexe.
- Criterii de acceptare: numerotare secvențială (AC1, AC2.1 etc.), scrise în sintaxă Gherkin, cu cazuri negative asociate celor pozitive corespunzătoare.

Cadrul sporește claritatea pentru dezvoltatori, testeri și stakeholderi prin unificarea cerințelor de business, funcționale și tehnice într-un singur artefact.

5.2 Validarea prin practică și training

Formatul EUS a evoluat prin aplicare iterativă în contexte profesionale și educaționale. În programul Ready for IT, peste 60 de absolvenți au testat formatul, dintre care mai mult de 80% s-au angajat ulterior ca BA sau PO. Multe companii au adoptat EUS intern, raportând timpi de rafinare reduși și predictibilitate mai bună în livrare. Angajatorii au descris EUS drept „un nou limbaj pentru specificarea cerințelor”.

5.3 Către o standardizare asistată de instrumente

Deși EUS a îmbunătățit claritatea, aplicarea manuală în backloguri mari a rămas consumatoare de efort. Pentru a aborda această problemă, a fost dezvoltată o extensie Jira:

- Secțiuni predefinite pentru overview, descriere și criterii de acceptare.
- Câmpuri care impun numerotarea secvențială și sintaxa Gherkin.
- Formatare integrată pentru elemente UI, mesaje de eroare și mesaje informaționale.
- Integrare în workflow pentru a semnaliza story-uri incomplete sau neconforme înainte de intrarea în sprint.

Extensia a redus efortul manual, a sprijinit onboardingul și a accelerat pregătirea backlogului în proiecte mari (de exemplu, platforme de e-commerce). Ea a demonstrat valoarea automatizării deterministe – consistență prin șabloane impuse, nu prin generare liberă.

5.4 De la șabloane la generare automată

În ciuda îmbunătățirilor, structurarea manuală a continuat să consume resurse. Evoluția EUS a trecut prin trei etape:

- Extensia Jira – a încorporat reguli deterministe direct în uneltele de backlog.
- Interfața bazată pe prompturi – a extins inputuri minime (AS A / I WANT TO / SO THAT) în story-uri EUS complet formate, aplicând automat regulile.
- Sistemul discovery-to-backlog – o aplicație proiectată să poată prelua inputuri eterogene (documente, transcrieri, prezentări) și să folosească NLP pentru a extrage actori, obiective și constrângeri. Acestea sunt grupate în epicuri și descompuse în story-uri generate automat în format EUS.

Această abordare stratificată ilustrează trecerea de la redactare manuală la impunere prin șabloane, generare semi-automată și la descoperire de cerințe bazată pe AI, având standardul EUS drept fundament care asigură consistență, claritate și testabilitate în toate etapele.

Capitolul 6. Extensia Jira pentru automatizarea deterministă a scrierii user story-urilor

Acest capitol descrie dezvoltarea și validarea Specs-Assistant, o extensie Jira creată pentru a încorpora standardul Extended User Story (EUS) direct în fluxurile Agile. Se evidențiază motivația, designul, funcționalitatea și beneficiile practice ale extensiei ca instrument de automatizare deterministă a scrierii user story-urilor.

6.1 Jira ca platformă pentru managementul cerințelor Agile

Jira este utilizată pe scară largă pentru managementul backlog-ului și al sprinturilor datorită configurabilității sale și integrării workflow-urilor, tipurilor de issue și câmpurilor personalizate. Ea oferă un depozit central de cerințe, dar nu impune convenții stricte pentru formatarea user story-urilor, ceea ce duce adesea la inconsistență și ambiguitate. Specs-Assistant abordează acest gol prin integrarea standardului EUS direct în Jira.

6.2 Motivația dezvoltării extensiei Jira

Aplicarea manuală a regulilor EUS presupune formatare și structurare repetitivă, care consumă aproape 20% din timpul de scriere a unui story. Extensia a fost motivată de două obiective:

- eficiență – reducerea timpului petrecut pe formatare repetitivă,
- consistență – asigurarea conformității cu regulile EUS pentru toate elementele din backlog.

6.3 Designul și funcționalitatea Specs-Assistant

Specs-Assistant se integrează în interfața Jira, oferind șabloane predefinite și funcții de validare. Principalele capabilități includ:

- Crearea de șabloane: inserarea automată a celor trei secțiuni EUS (overview, descriere, criterii de acceptare).
- Criterii de acceptare pre-formate: numerotare automată (AC1, AC2.1 etc.) cu cazuri limită și negative explicite.
- Impunerea regulilor de formatare: elementele UI, mesajele de eroare și răspunsurile sistemului sunt stilizate conform regulilor EUS.
- Suport pentru validare: împiedicarea salvării story-urilor incomplete sau neconforme.

- Ușurință în utilizare: integrat în fluxul nativ Jira, cu o curbă de învățare minimă.

6.4 Interfața extensiei

Extensia Specs-Assistant a fost concepută având în vedere ușurința de utilizare, integrându-se perfect în fluxul nativ Jira. Interfața are scopul de a reduce efortul cognitiv și operațional al scrierii user story-urilor, ghidând analiștii pas cu pas printr-un proces standardizat.

După instalare, extensia activează un tablou principal care servește ca spațiu central de lucru pentru Business Analyst și Product Owner. Acest tablou oferă o privire structurată asupra tuturor epicurilor și story-urilor asociate, asigurând că cerințele sunt organizate ierarhic și consistent. Din acest tablou, utilizatorii pot:

- crea elemente noi de backlog direct cu secțiuni EUS predefinite (overview, descriere, criterii de acceptare),
- naviga vizual printre epicuri și story-uri, reducând riscul de a omite dependențe,
- vizualiza relațiile dintre epicuri și story-urile aferente, facilitând trasabilitatea.

Figura 6.4.3 ilustrează această funcționalitate, arătând cum extensia afișează epicurile la nivel superior și le leagă de story-urile corespunzătoare. Această vizualizare ierarhică ajută la menținerea alinierii între obiectivele de business și cerințele detaliate.

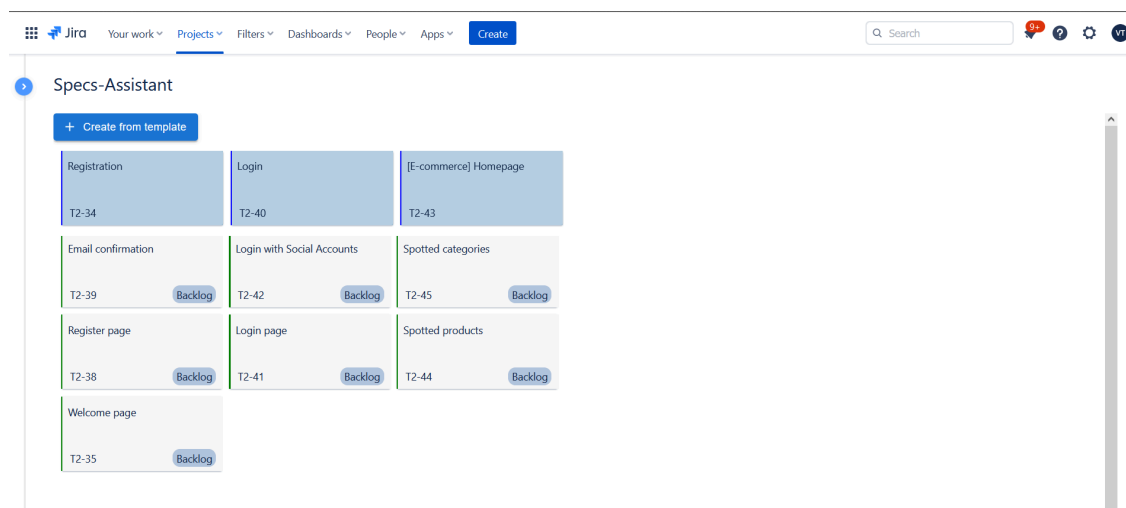


Figure 6.4.3. Dashboard with Epics and Stories

La crearea sau editarea unui user story, interfața deschide o fereastră Jira îmbogățită cu funcții specifice EUS:

- toate secțiunile obligatorii (AS A / I WANT TO / SO THAT, descriere, criterii de acceptare) sunt pre-populate ca placeholders, prevenind omisiunile,
- câmpuri personalizate (de ex. identificatori de reglementare, taguri de trasabilitate) pot fi impuse, asigurând conformitatea cu guvernanta organizațională,
- regulile de formatare pentru elemente UI, mesaje de eroare și răspunsuri ale sistemului sunt aplicate automat, eliminând munca manuală repetitivă,
- verificări de validare blochează salvarea story-urilor incomplete sau structurate incorect; de exemplu, dacă lipsesc criteriile de acceptare sau nu sunt numerotate corect, sistemul solicită completarea înainte de a permite salvarea.

Această experiență ghidată aduce multiple avantaje:

- analiștii evită sarcinile de formatare repetitivă,
- testerii și dezvoltatorii beneficiază de structuri uniforme ale story-urilor,
- managerii obțin o calitate mai predictibilă și trasabilă a backlog-ului.

Transformând redactarea story-urilor într-un flux bazat pe șabloane și validări, Specs-Assistant reduce efortul și garantează că toate elementele backlog-ului respectă Definition of Ready.

6.5 Beneficiile observate în practică

Specs-Assistant a adus beneficii clare: reducerea timpului de scriere a story-urilor cu 15–20%, respectarea constantă a standardelor structurale, rafinare mai rapidă și onboarding mai ușor pentru membrii noi ai echipei. Prin încorporarea șabloanelor EUS direct în Jira, extensia a operaționalizat standardizarea și a marcat primul pas către o automatizare mai largă a ingineriei cerințelor, evoluând de la impunerea deterministă la generare inteligentă și transformarea directă discovery-to-backlog.

Capitolul 7 – Interfața AI pentru generarea user story-urilor

Acest capitol explorează tranziția de la abordările deterministe, bazate pe reguli, la automatizarea prin inteligență artificială în generarea user story-urilor. El evidențiază aplicarea API-ului GPT-3 de la OpenAI în ingineria cerințelor, subliniind eficiența, scalabilitatea și adaptabilitatea în practicile Agile.

Motivația derivă din limitele redactării manuale: user story-urile sunt adesea inconsistente, ambigue și consumatoare de timp. În timp ce metodele deterministe, cum ar fi o extensie Jira concepută pentru proiecte de e-commerce, au arătat rezultate încurajatoare (până la 60% reducere a efortului de scriere), dependența lor de șabloane rigide a limitat adaptabilitatea în alte domenii [5]. Modelele de tip Large Language Models (LLMs), și în special GPT-3, oferă o soluție mai flexibilă și scalabilă, transformând inputuri nestructurate în cerințe structurate, aliniate la standardul Extended User Story (EUS).

7.1 Lucrări conexe

Mai multe platforme și studii au avansat automatizarea generării user story-urilor:

- Atlassian Jira a introdus funcționalități AI precum recomandarea de formate, generarea automată de criterii de acceptare și detectarea inconsistențelor.
- AgileGen [29] se concentrează pe cerințe testabile bazate pe Gherkin pentru a asigura alinierea între specificații și cod, deși abordează etape ulterioare ale ciclului de viață comparativ cu acest studiu.
- Sisteme de AI generativă precum ChatGPT și Claude au fost folosite tot mai des pentru a produce versiuni preliminare de story-uri și pentru a le rafina, oferind câștiguri de eficiență și outputuri mai sigure, adaptate contextului.

Cercetările recente subliniază lipsa unor seturi mari și deschise de date pentru antrenarea unor modele robuste și evidențiază importanța prompt engineering-ului ca strategie pentru obținerea unor rezultate consistente de la modelele AI [32]. Studii mai ample privind AI în managementul de proiect Agile scot în evidență beneficii precum îmbunătățirea gestionării backlog-ului, colaborării și reducerii riscurilor [5].

7.2 De la șabloane manuale la generare asistată de AI

Primul efort de automatizare s-a bazat pe o extensie deterministă Jira. Ea încorporează șabloane predefinite alcătuite din:

- persona, obiectivul utilizatorului și motivația,
- descrieri detaliate ale funcționalităților,
- criterii de acceptare în format Gherkin.

Deși eficientă în e-commerce, această metodă nu era generalizabilă. Șabloanele trebuiau recreate manual pentru fiecare domeniu, ceea ce limita scalabilitatea [5].

Abordarea bazată pe reguli era specifică domeniului. Diferite industrii (sănătate, finanțe, retail) necesită terminologii, personaje și factori de conformitate distincte. Acest fapt a scos în evidență limitele automatizării deterministe și a motivat trecerea spre generare asistată de AI [5].

Prin integrarea API-ului GPT-3 de la OpenAI, cercetarea a introdus o automatizare care putea fi scalată între domenii. Cu ajutorul prompt engineering-ului, GPT-3 a generat story-uri conforme cu EUS pornind de la inputuri minime, asigurând claritate și adaptabilitate.

Figura 7.2.1 arată cum Business Analystul oferă doar tipul de utilizator și acțiunea dorită, iar asistentul generează un story complet.

The screenshot displays the 'BA-Assistant' interface. On the left, a form titled 'Create your user story' has two input fields. The first field, labeled 'AS A', contains the text 'visitor'. Below it, 'examples:' are listed as 'visitor', 'registered user', and 'customer'. The second field, labeled 'I WANT TO', contains the text 'access a login page'. Below it, 'examples:' are listed as 'access a login page' and 'send an order'. A purple 'Generate' button is positioned below the form. On the right, the generated user stories are displayed in a light gray box. They follow the Gherkin syntax: 'AS A visitor', 'I WANT TO access a login page', and 'SO THAT I can log in to the website and access personalized content or services.' Below this, three 'USER ACCEPTANCE CRITERIA' are listed, each with a title, a 'GIVEN' condition, a 'WHEN' condition, and a 'THEN' result. The first criterion is for successful login, the second for redirection after login, and the third for an error message on failed login. A purple 'Copy' button is at the bottom of the generated text area.

Create your user story

AS A

examples: [visitor](#) [registered user](#) [customer](#)

I WANT TO

examples: [access a login page](#) [send an order](#)

Generate

AS A visitor
I WANT TO access a login page
SO THAT I can log in to the website and access personalized content or services.

USER ACCEPTANCE CRITERIA
AC1: Title: Visitor can access the login page
GIVEN: The visitor is on the website homepage
WHEN: The visitor clicks on the "Login" button or navigates to the login page URL
THEN: The login page is displayed

AC2: Title: Visitor cannot access the login page if already logged in
GIVEN: The visitor is already logged in
WHEN: The visitor clicks on the "Login" button or navigates to the login page URL
THEN: The visitor is redirected to their account page or dashboard

AC3: Title: Visitor cannot access the login page with invalid credentials
GIVEN: The visitor is on the login page
WHEN: The visitor enters invalid login credentials (e.g., incorrect username or password)
THEN: An error message is displayed, stating that the login was unsuccessful

Copy

Figure 7.2.1. Interfața client BA-Assistant

Figura 7.2.2 ilustrează arhitectura: interfața client, serverul aplicației, modelele de embedding și motorul GPT interacționează pentru a procesa inputurile și a genera răspunsuri.

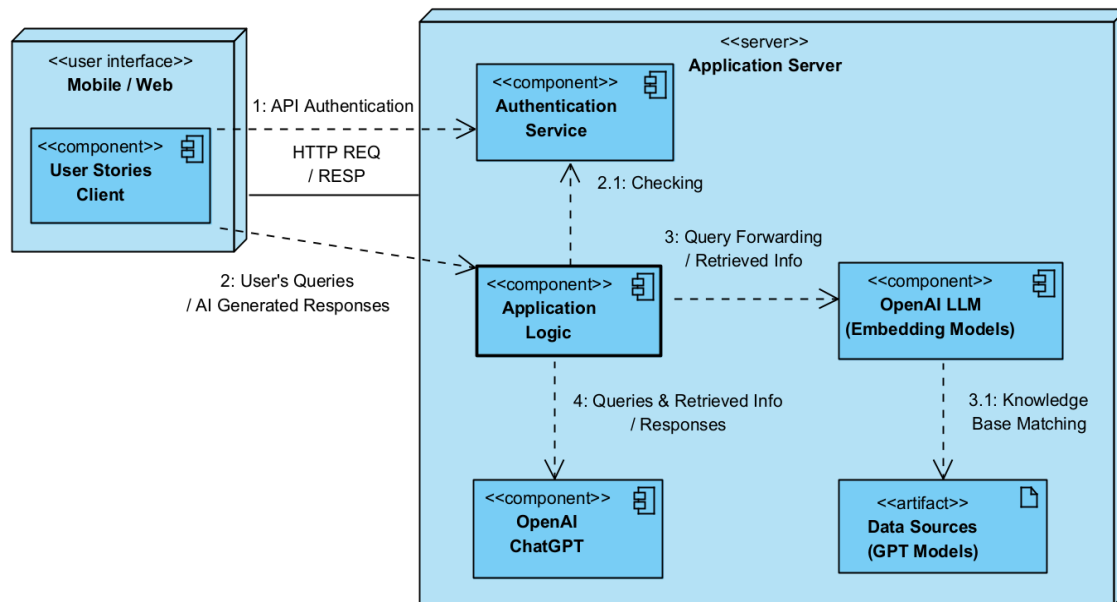


Figure 7.2.2. Arhitectura de nivel înalt a prototipului inițial bazat pe inteligență artificială

Prompt engineering-ul a aliniat outputurile AI la practicile Agile prin utilizarea de exemple structurate și testări iterative, așa cum este prezentat în Figura 7.2.3.

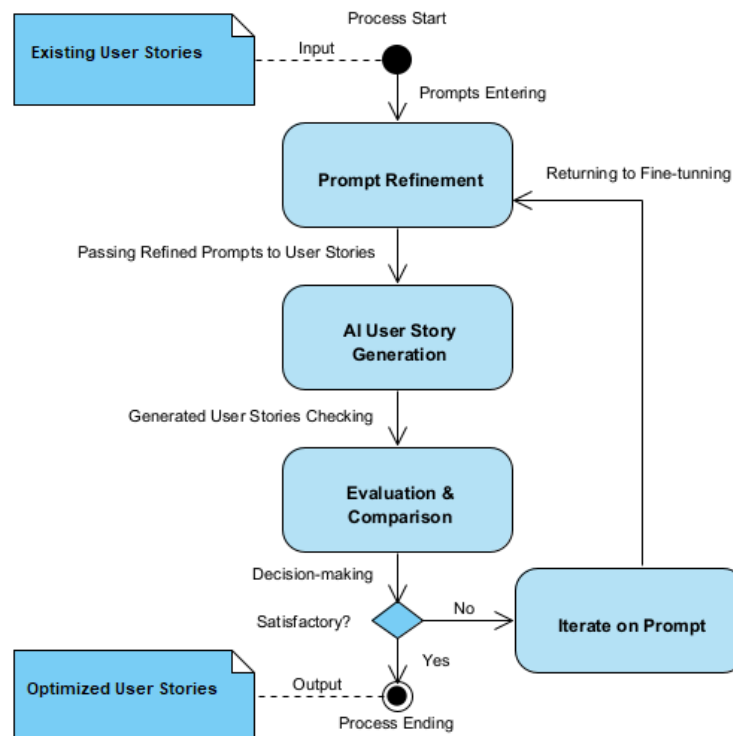


Figure 7.2.3. Fluxul de Prompt Engineering

7.3 Evaluarea comparativă a modelelor AI

Au fost testate două modele: ChatGPT și Gemini, folosind prompturi echivalente.

Tabelul 7.2. Evaluarea comparativă a modelelor AI (ChatGPT vs. Gemini).

Model	Puncte forte	Puncte slabe
ChatGPT	Înțelegere contextuală puternică; criterii de acceptare detaliate; aliniere mai bună cu valoarea de business	Uneori prea verbose
Gemini	Outputuri concise, clare și bine structurate	Mai puțin detaliate; uneori omite criterii de acceptare

Ambele au produs story-uri coerente și conforme cu EUS, dar ChatGPT a demonstrat o profunzime contextuală superioară în cazurile complexe [5].

7.4 Rezultate și provocări

Generarea de story-uri asistată de AI aduce îmbunătățiri semnificative în ingineria cerințelor prin reducerea efortului manual, asigurarea scalabilității între industrii, menținerea consistenței cu standardul EUS și integrarea capacităților de învățare prin feedback-ul utilizatorilor.

În același timp, trebuie abordate mai multe provocări, printre care: protejarea cerințelor sensibile pentru a asigura confidențialitatea datelor, oferirea de explicații clare pentru ca stakeholderii să poată înțelege și avea încredere în outputurile AI și menținerea controlului uman pentru a garanta acuratețea contextuală și alinierea la nevoile stakeholderilor.

Capitolul concluzionează că AI schimbă definirea user story-urilor dintr-o activitate manuală repetitivă într-un proces scalabil și adaptabil, oferind suport mai puternic echipelor Agile și proiectelor la scară de întreprindere.

Capitolul 8. Descoperirea multimodală a cerințelor asistată de AI

Capitolul introduce Discovery AI, o platformă bazată pe inteligență artificială care abordează ambiguitatea, lipsa de comunicare și trasabilitatea slabă din fazele timpurii ale descoperirii Agile, prin integrarea naturii multimodale și conversaționale a întâlnirilor reale. În loc să trateze cerințele ca artefacte strict textuale, Discovery AI procesează înregistrări audio, note și indicii vizuale și – prin ASR, RAG și LLM-uri precum GPT-4o, Llama 3.2 ajustat și Gemini 2-Flash – produce cerințe Agile structurate (epicuri, user story-uri, teme).

Dezvoltat iterativ, primul prototip complet (publicat deja în lucrări academice) arată îmbunătățiri măsurabile privind claritatea și trasabilitatea cerințelor, acuratețea clasificării și viteza documentării, în timp ce o variantă simplificată este discutată în secțiunile ulterioare [26]. Prin poziționarea întâlnirilor ca surse dinamice de cunoaștere și prin combinarea datelor reale cu cele sintetice, Discovery AI contrazice presupunerea anterioară că cerințele sunt exclusiv textuale sau deterministe și demonstrează adaptabilitate cross-industrie [27].

8.1 Tehnologii fundamentale în descoperirea cerințelor asistată de AI

- ASR (Automatic Speech Recognition). Strat modular care convertește audio din întâlniri în transcrieri sincronizate pe timp și pe vorbitori (cu reducerea zgomotului, diarizare și segmentare), permițând extragerea robustă chiar și cu accente, suprapuneri sau dialog informal.
- RAG (Retrieval-Augmented Generation). Compune răspunsuri prin regăsirea contextului relevant dintr-o bază vectorială și injectarea lui în prompturi înainte de generare. În Discovery AI este utilizat pentru rafinare, generarea de user story-uri și clasificare, reducând halucinațiile și asigurând continuitatea între proiecte similare.
- LLM-uri. Discovery AI combină GPT-4o pentru înțelegere inițială și redactare, Llama 3.2 ajustat pentru rigoare structurală și aliniere la Agile, și Gemini 2-Flash pentru sumarizare rapidă. Prompturile sunt gestionate în Langfuse, care permite control prin versionare, A/B testing și monitorizarea performanței.
- Infrastructură de suport. Qdrant pentru căutare semantică rapidă în întâlniri, user story-uri, literatură și seturi de date sintetice; Langfuse pentru versionarea prompturilor și trasabilitate; model registry pentru integrarea ușoară în unelte precum Jira sau Notion.

8.2 De la unelte fragmentate la fluxuri integrate de descoperire

În mod obișnuit, lanțurile de unelte sunt fragmentate (transcriere separată, sumarizare separată, mapare de story-uri, documentare). Discovery AI integrează pipeline-ul complet, de la input multimodal brut (audio, video, note) la documentație Agile structurată (epicuri și user story-uri), fără a necesita prestructurare sau expertiză tehnică din partea utilizatorului.

8.3 Abordarea cu două faze de generare

Discovery AI separă redactarea de rafinare pentru a combina viteza cu rigoarea.

- GPT-4o procesează transcrieri ASR segmentate și diarizate, sumarizări cu Gemini 2-Flash și cunoștințe regăsite prin RAG, generând schițe de epicuri și user story-uri de tip EUS – rapide, creative și generale.
- Llama 3.2 rafinează aceste schițe, aplică șablonul Agile („Ca..., vreau..., pentru a...“), adaugă terminologie specifică domeniului, elimină redundanțele și armonizează story-urile la nivel de epic, obținând artefacte gata de producție.

Tabelul 8.3 rezumă diferențele: GPT-4o asigură viteză și acoperire largă, dar are risc mai mare de halucinații, în timp ce Llama 3.2 aduce rigoare, acuratețe și conformitate Agile.

Tabelul 8.3. Redactarea cu GPT-4o și rafinarea cu Llama 3.2

Caracteristică	Redactare cu GPT-4o (Faza 1)	Rafinare cu Llama 3.2 (Faza 2)
Viteză	Foarte ridicată	Moderată (analiză mai profundă)
Acoperirea contextului	Acoperire largă a inputurilor	Focus specific domeniului
Conformitate cu Agile	Parțială (necesită validare)	Conformitate completă cu șablonul
Risc de halucinații	Ridicat fără RAG	Redus (validare prin rafinare)
Calitatea outputului	Grosieră, exploratorie	Precisă, gata de producție

Acest design în două etape asigură că outputurile sunt fundamentate, generate eficient, aliniate la principiile Agile și rafinate pentru coerență și trasabilitate.

8.4 Prezentare generală a sistemului

Discovery AI integrează trei procese: construirea bazei de cunoștințe, înțelegerea întâlnirilor și rafinarea cu LLM.

- Baza de cunoștințe include literatură de specialitate, documente organizaționale, transcrieri și date sintetice indexate în Qdrant, care alimentează RAG.
- Înțelegerea întâlnirilor presupune transcrierea audio/video cu Whisper ASR, sumarizarea cu Gemini 2-Flash și analiza semantică cu GPT-4o.
- Rafinarea este realizată prin Llama 3.2, ajustat pe seturi de date curate și versionate, cu deployment prin Hopsworks Model Registry.

Astfel, cerințele generate sunt fundamentate pe informații verificate și relevante, reducând halucinațiile și menținând acuratețea. Spre deosebire de unele limitate la transcrieri, Discovery AI integrează surse diverse pentru regăsire semantică, asigurând alinierea la bune practici și la istoricul proiectelor.

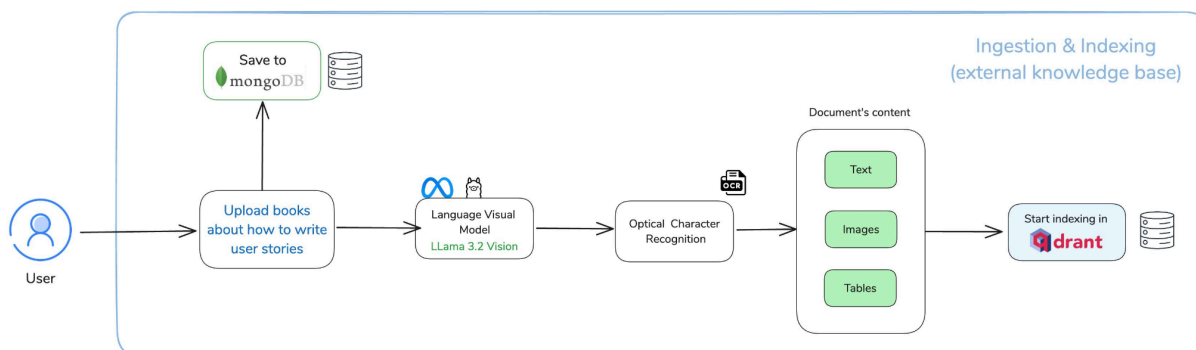


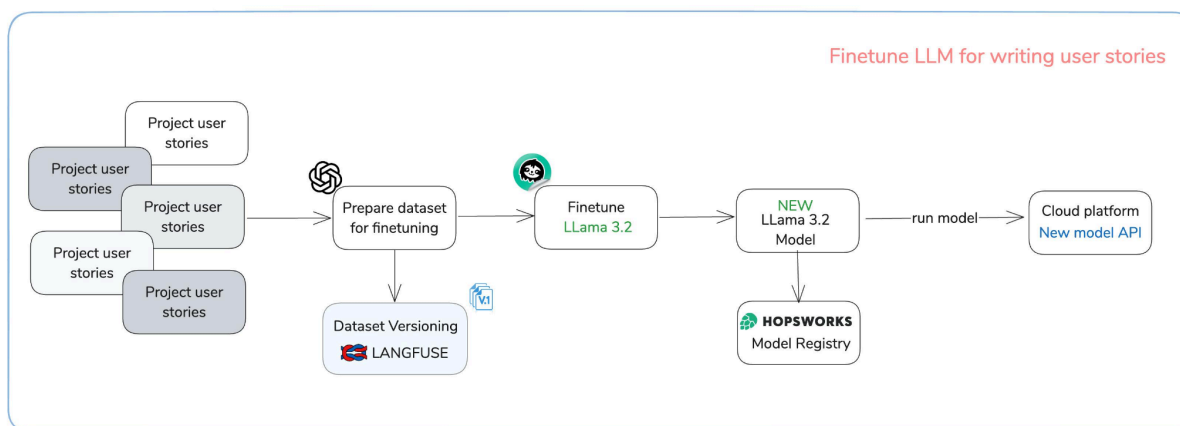
Figura 8.4.1. Fluxul bazei externe de cunoștințe în Discovery AI

Întâlnirile de business sunt surse bogate, dar nestructurate de cerințe. Modulul de Înțelegere a Întâlnirilor din Discovery AI convertește înregistrările audio/video brute în informații structurate și ușor de regăsit. Înregistrările sunt încărcate în Amazon S3, asigurând durabilitatea și posibilitatea de reprocesare. Ele sunt transcrise de Whisper ASR, ales pentru robustețea sa în contexte multilingve, zgomotoase și cu vorbire suprapusă. Transcrierile, îmbogățite cu marcate de timp și diarizare pe vorbitori, sunt stocate în MongoDB pentru trasabilitate.

În continuare, transcrierile sunt procesate de GPT-o1 (GPT-4o), care extrage sensul latent și cerințele candidate, ghidat de șabloane de prompturi versionate în Langfuse. O cale paralelă de sumarizare cu Gemini 2-Flash produce sinteze concise ale întâlnirilor, permițând indexare rapidă

Împreună, Qdrant (căutare semantică după similaritate) și MongoDB (metadate structurate) formează o strategie de stocare pe două straturi: Qdrant permite regăsirea bazată pe concepte, în timp ce MongoDB asigură trasabilitatea. Prin acest pipeline hibrid, ilustrat în Figura 8.4.2, Discovery AI transformă întâlnirile nestructurate în artefacte Agile structurate, fundamentate atât pe date reale, cât și sintetice.

încorporează un pipeline dedicat de fine-tuning bazat pe Llama 3.2, susținut de pregătirea sistematică a dataseturilor, versionare și mecanisme de implementare.



Fluxul de fine-tuning pentru scrierea user story-urilor în Discovery AI

8.5 Fluxul de inferență

Fluxul de inferență reprezintă faza operațională a Discovery AI, în care sistemul este utilizat de către utilizatorii finali (de exemplu, product manageri, product owneri sau business analiști) pentru a genera epicuri și user story-uri pentru un proiect dat. Spre deosebire de pipeline-urile de antrenare și fine-tuning, care sunt în mare parte procese offline, fluxul de inferență funcționează aproape în timp real și trebuie să echilibreze eficiența, acuratețea și fundamentarea contextuală. Procesul este ilustrat în Figura 8.5 și se desfășoară în patru etape secvențiale:

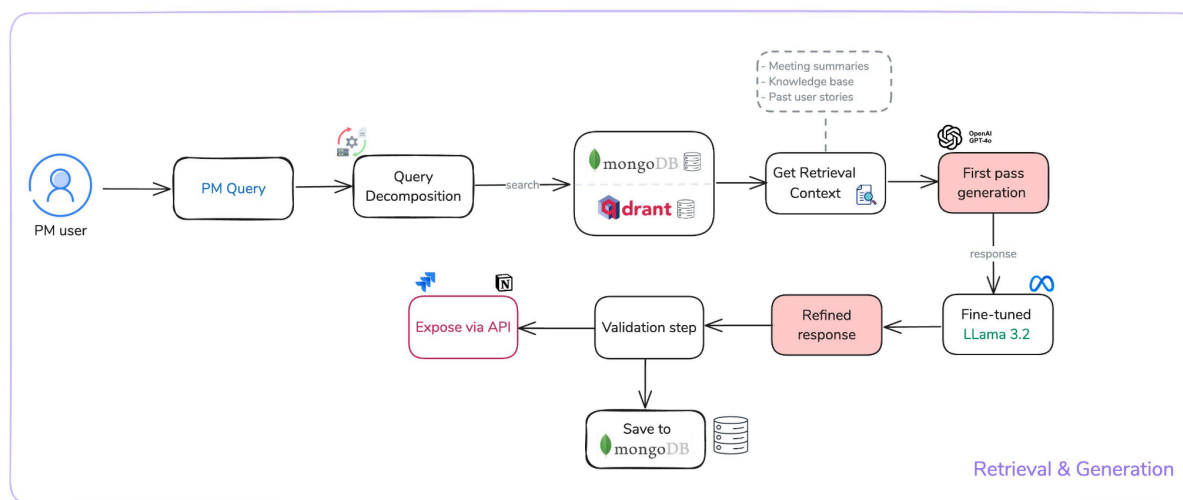


Figure 8.5. Fluxul de inferență al Discovery AI

Capitolul 9: O iterație simplificată a Discovery AI cu GPT-5

Acest capitol introduce a doua iterație a Discovery AI, reproiectată pentru a valorifica capabilitățile multimodale și de raționament ale GPT-5. Prima versiune se baza pe un pipeline complex, multi-model, care includea Whisper pentru transcriere, GPT-4o pentru generarea primei versiuni, Gemini 2-Flash pentru sumarizare și Llama 3.2 pentru rafinare. Deși această abordare era funcțională, ea introducea latență semnificativă, costuri ridicate de mentenanță și o complexitate considerabilă a integrării.

Odată cu lansarea GPT-5, care integrează recunoaștere vocală, sumarizare, raționament și generare conformă cu Agile într-un singur model, a devenit posibil un pipeline simplificat. După cum se arată în Figura 9.1, noul sistem consolidează funcționalități care anterior erau distribuite între mai multe componente, reducând dependențele și îmbunătățind robustețea.

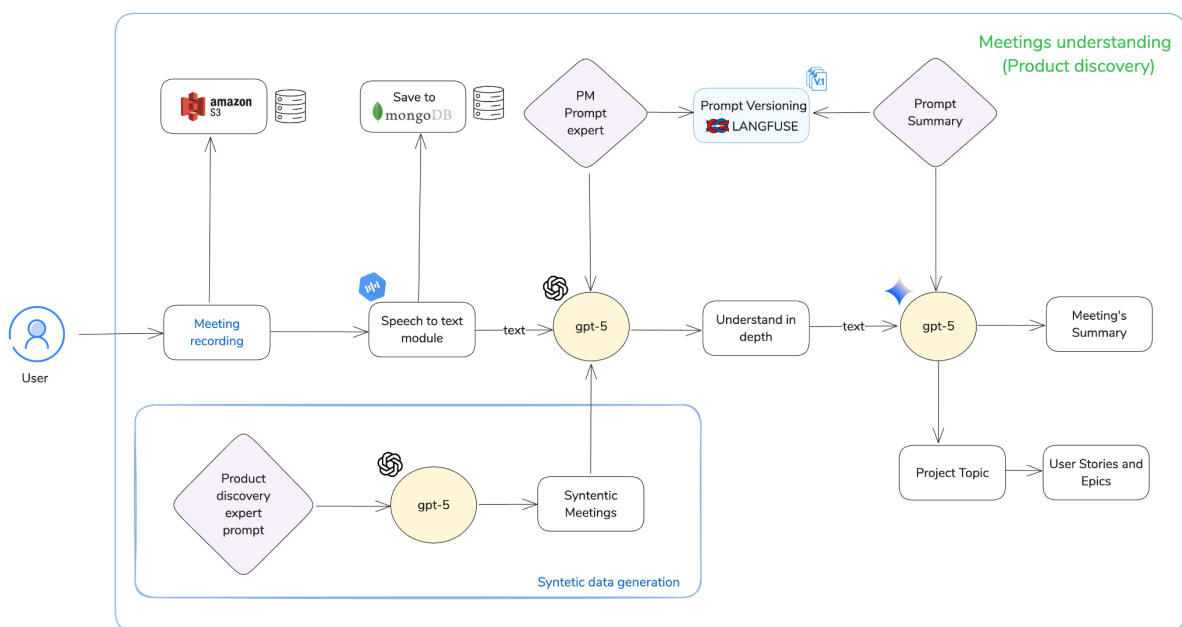


Figura 9.1. Discovery AI simplificat cu GPT-5 – de la înregistrările întâlnirilor la user story-uri și epicuri într-un flux de lucru eficientizat.

9.1 Designul sistemului

Pipeline-ul simplificat începe cu notele și înregistrările întâlnirilor, stocate în Amazon S3 pentru durabilitate și reproductibilitate. Transcrierile procesate, sumarizările și artefactele Agile sunt păstrate în MongoDB, asigurând auditabilitate. GPT-5 gestionează nativ transcrierea, eliminând nevoia de Whisper și, simultan, realizează îmbogățirea semantică, captând cerințele implicite în

timpul transcrierii. În plus, integrează raționament, sumarizare și generare structurată, producând direct user story-uri și epicuri în formatul Agile canonic, complete cu criterii de acceptare.

Managementul prompturilor rămâne esențial, deși aria sa este redusă. Cu ajutorul Langfuse, prompturile sunt versionate și optimizate pentru a asigura reproductibilitatea și consistența outputurilor. În loc să orchestreze mai multe modele, Langfuse interacționează acum în principal cu GPT-5, concentrându-se pe extragere structurată, clasificare tematică și generare de user story-uri Agile.

Trei tipuri de prompturi structurate sunt centrale pentru a asigura consistența și utilitatea outputurilor:

- primul prompt impune un schelet JSON pentru extragerea insighturilor cheie din întâlniri, cum ar fi puncte de discuție, cerințe, riscuri și decizii, făcându-le direct lizibile de mașină;
- al doilea organizează discuțiile în clasificări tematice procentuale, cuantificând cât dintr-o întâlnire a fost dedicat unor categorii predefinite precum magazin online, rezervări sau programări;
- al treilea definește formatul pentru generarea de epicuri și user story-uri, cerând structuri conforme cu Agile și criterii de acceptare scrise în sintaxă Gherkin, pentru ca outputurile să poată fi integrate imediat în fluxurile de lucru de management al proiectelor.

9.2 Analiză comparativă: Iterația veche vs. nouă

Simplificarea obținută prin GPT-5 este cel mai bine ilustrată printr-o comparație între cele două iterații ale Discovery AI. Tabelul 9.1 rezumă principalele diferențe.

Tabelul 9.1. Comparație între Discovery AI v1 (pipeline multi-model) și v2 (simplificat cu GPT-5)

Caracteristică / Dimensiune	Discovery AI v1 (multi-model)	Discovery AI v2 (simplificat cu GPT-5)
Recunoaștere vocală	Whisper ASR (modul separat)	Procesare multimodală nativă GPT-5
Sumarizare	Gemini 2-Flash	Sumarizare integrată în GPT-5

Generarea primei versiuni	GPT-4o	GPT-5
Rafinare	Llama 3.2 ajustat	GPT-5 (fără rafinare separată necesară)
Regăsirea cunoștințelor (RAG)	Qdrant + MongoDB	Qdrant + MongoDB (cu overhead redus)
Generarea de date sintetice	GPT-4o ghidat prin prompturi	GPT-5 direct, mai integrat
Latența medie per proiect	2–3 minute	< 30 secunde (estimat)
Complexitatea mentenanței	Ridicată (mai multe modele și pipeline-uri)	Redusă (un singur model de bază)

Această comparație evidențiază compromisurile dintre modularitatea detaliată (v1) și simplitatea integrată (v2).

9.3 Interfața și utilizabilitatea

Discovery AI v2 introduce, de asemenea, o interfață reproiectată (Figura 9.3), care integrează toate etapele fluxului de lucru într-un singur tablou de bord. Utilizatorii pot încărca fișierele întâlnirilor, pot inspecta insighturile structurate, pot revizui clasificările tematice și pot accesa user story-urile generate de AI, împreună cu criteriile de acceptare. Aceste outputuri pot fi editate și exportate direct în platforme de management de proiect precum Jira sau ClickUp. Această accesibilitate reflectă valorile Agile de transparență și colaborare, permițând atât stakeholderilor tehnici, cât și celor non-tehnici să se implice în cerințele generate de AI.

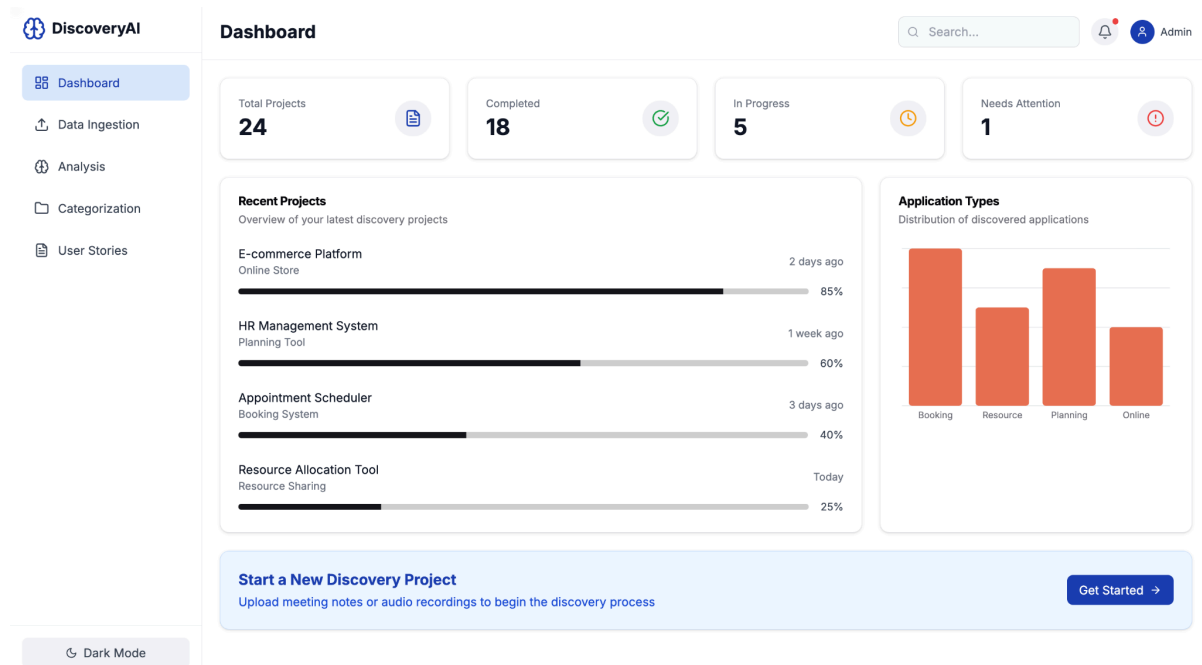


Figura 9.3. Interfața Discovery AI - Privire de ansamblu asupra tabloului de bord

Interfața revizuită din Discovery AI v2 abordează aceste neajunsuri, permițând interacțiunea directă cu toate etapele pipeline-ului printr-un singur tablou de bord (vezi Figura 9.4.).

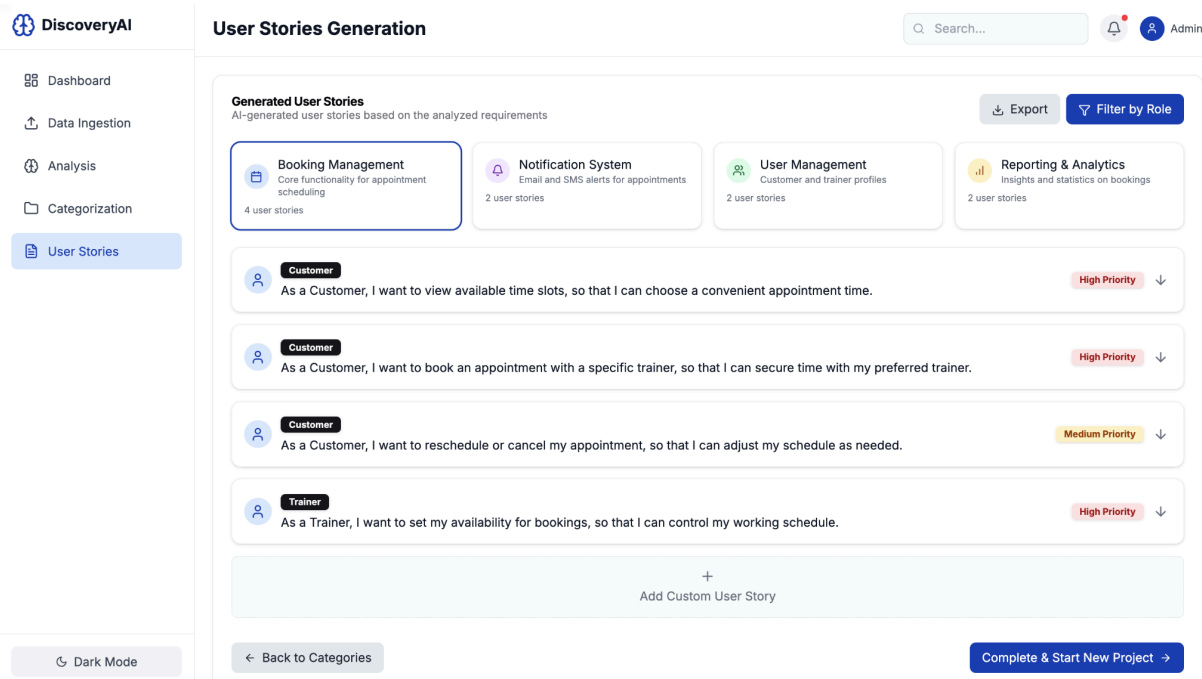


Figura 9.4. Interfața Discovery AI – Epicuri și user story-uri generate cu criterii de acceptare

9.4 Oportunități și limitări

Deși pipeline-ul simplificat oferă câștiguri clare de eficiență și utilizabilitate, el introduce și noi dependențe. Sistemul se bazează acum în totalitate pe GPT-5; orice limitări sau biasuri ale modelului afectează toate etapele fluxului de lucru. Mai mult, eliminarea unei etape separate de rafinare reduce oportunitățile de personalizare specifică domeniului. Cercetările viitoare ar putea explora strategii hibride care să păstreze eficiența GPT-5, dar să includă și componente ajustate fin, ușoare, pentru contexte specializate.

Capitolul 10. Concluzii

10.1 Concluzii generale

Această cercetare a abordat provocarea critică a definirii cerințelor prin introducerea unor soluții standardizate, semi-automatizate și îmbunătățite cu AI, menite să sporească claritatea, trasabilitatea și eficiența. Pe baza dovezilor conform cărora cerințele neclare determină peste 80% dintre eșecurile proiectelor, a fost propus standardul Extended User Story (EUS) și s-a evoluat de la o extensie Jira deterministă la un prototip multimodal bazat pe AI, capabil să genereze user story-uri din transcrieri, înregistrări și documente. Experimentele au confirmat fezabilitatea, scalabilitatea și valoarea practică a acestor instrumente atât în contexte corporative, cât și educaționale.

10.2 Contribuții originale

Cercetarea doctorală a adus multiple contribuții originale, atât teoretice, cât și practice:

- Contribuții teoretice și metodologice:
 - Definirea formatului Extended User Story (EUS), o nouă structură de user story care combină o prezentare generală, o descriere detaliată și criterii de acceptare în format Gherkin, împreună cu marcaje vizuale ce sporesc claritatea și utilizabilitatea.
 - Introducerea regulilor de formatare care diferențiază elementele de interfață, mesajele sistemului și mesajele de eroare pentru reducerea ambiguității.
- Contribuții tehnice:
 - Dezvoltarea unei extensii JIRA care integrează structura EUS în fluxurile Agile și accelerează redactarea și validarea user story-urilor în proiecte reale.
 - Crearea unui prototip bazat pe AI care integrează API-ul OpenAI pentru a genera user story-uri pe baza prompturilor utilizatorilor sau a notelor din întâlniri, obținând până la 60% reducere a timpului în faza de definire a story-urilor.
 - Propunerea unei arhitecturi AI (în curs de publicare) capabile să extragă cerințe, structurate printr-un pipeline orchestrat.
- Contribuții științifice și educaționale:

- Publicarea a două articole peer-reviewed care descriu arhitectura propusă și tranziția de la ingineria deterministă la cea bazată pe AI.
- Integrarea și validarea formatului EUS în programul de training Ready for IT, unde peste 80% dintre absolvenți au obținut cu succes roluri ca Business Analysts sau Product Owners, dintre care unii au implementat standardul în companiile lor de angajare.

10.3 Direcții de cercetare viitoare

Cercetările viitoare ar trebui să se concentreze pe integrarea unor instrumente precum Jira și Confluence pentru o automatizare completă, ajustarea fină a modelelor GPT pe seturi de date precum NeoDataset pentru o mai bună adaptare la domeniu și extinderea Discovery AI pentru a procesa interacțiuni verbale. Direcții suplimentare includ dezvoltarea unor agenți de validare pentru consistență și conformitate, implementarea unor module de explicabilitate și siguranță pentru industrii reglementate și testarea sistemului în domenii precum sănătate, fintech, retail și servicii publice, pentru a asigura robustețea.

10.4 Sinteza lucrării

Cercetarea doctorală a urmărit în mod constant descoperirea cerințelor, automatizarea și integrarea AI în ingineria software Agile, combinând inovația teoretică cu validarea practică în industrie și mediul academic. Contribuțiile de bază includ două articole indexate ISI: „Transforming user story definition: From deterministic to AI-powered automation” și „Enhancing agile requirement discovery with AI-driven multimodal systems and synthetic user story generation”, care stau la baza Capitolului 7 și Capitolului 8.

Rezultatele suplimentare includ publicații BDI privind microserviciile AI, selecția echipelor Scrum și platforma Discovery AI, prezentate la conferințe Smart Cities. La nivel profesional, cercetarea a fost extinsă prin formarea a peste 80 de specialiști prin Academia Ready for IT, cu o rată ridicată de angajare, și prin dezvoltarea de conținut educațional ulterior integrat în Capitolele 5 și 6. Autoarea a susținut, de asemenea, cursuri academice, a organizat evenimentul „Life in Tech” și a participat activ la conferințe internaționale majore, asigurând un feedback continuu din partea practicienilor.

Recunoașterea în Forbes 30 Under 30 România (2023) ca co-fondator al Genezio subliniază și mai mult intersecția dintre cercetare și impactul antreprenorial.

În ansamblu, teza stabilește o fundație metodologică și tehnologică pentru avansarea descoperirii cerințelor software bazate pe AI, validată în contexte corporative, academice și interdisciplinare.

Referințe

- [1] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.
- [2] Standish Group, *CHAOS Report*, 2020.
- [3] M. Cohn, *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.
- [4] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, “The Use and Effectiveness of User Stories in Practice,” *Requirements Engineering*, vol. 21, no. 3, pp. 383–403, 2016.
- [5] **A.-P. Avasiloaie**, A. Semenescu, E.-C. Popovici, “Transforming User Story Definition: From Deterministic to AI-Powered Automation,” *Romanian Journal of Information Technology and Automatic Control*, vol. 35, no. 2, pp. 59–72, 2025. WOS:001522929000005
- [6] B. Ramesh and L. Cao, “Agile Requirements Engineering Practices and Challenges,” *IEEE Software*, vol. 27, no. 6, pp. 60–67, 2010.
- [7] A. Ferrari, P. Spoletini, and S. Gnesi, “Ambiguity and Tacit Knowledge in Requirements Elicitation Interviews,” *Requirements Engineering Journal*, vol. 21, no. 3, pp. 333–355, 2016.
- [8] H. Hassani, X. Huang, and E. Silva, “Artificial Intelligence in Requirements Engineering: Opportunities and Challenges,” *Information Systems*, vol. 108, p. 102012, 2022.
- [9] N. Bennett and G. Lemoine, “What VUCA Really Means for You,” *Harvard Business Review*, 2014.
- [10] S. Kraus, T. Clauss, M. Breier, J. Gast, A. Zardini, and V. Tiberius, “The Economics of COVID-19: Initial Empirical Evidence on How Family Firms in Five European Countries Cope with the Corona Crisis,” *International Journal of Entrepreneurial Behavior & Research*, vol. 27, no. 2, 2021.
- [11] H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 13th ed. Wiley, 2022.
- [12] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner’s Approach*, 9th ed. McGraw-Hill, 2019.
- [13] B. Boehm, “A Spiral Model of Software Development and Enhancement,” *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1986.
- [14] K. Schwaber and J. Sutherland, *The Scrum Guide*, 2020.
- [15] W. W. Royce, “Managing the Development of Large Software Systems,” in *Proc. IEEE WESCON*, 1970.
- [16] K. Beck et al., *Manifesto for Agile Software Development*, 2001. [Online]. Available: <https://agilemanifesto.org/>
- [17] PMI & Digital.ai, *Agile Adoption Report*, 2024.
- [18] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley, 2010.

-
- [19] L. Leite, A. Rocha, C. Silva, T. Conte, and C. R. B. de Souza, "Large Language Models in Agile Practices: Opportunities and Risks," *Empirical Software Engineering*, vol. 29, 2024.
- [20] S. Denning, *The Age of Agile: How Smart Companies Are Transforming the Way Work Gets Done*. Amacom, 2018.
- [21] D. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.
- [22] Scaled Agile Inc., *SAFe Framework*, Version 6.0, 2023.
- [23] H. Sharp and H. Robinson, "Some Social Factors of Software Engineering: The Maturing of Agile Methods," *Information and Software Technology*, vol. 48, no. 6, pp. 541–552, 2006.
- [24] B. Ramesh, L. Cao, and R. Baskerville, "Agile Requirements Engineering Practices and Challenges: An Empirical Study," *Information Systems Journal*, vol. 20, no. 5, pp. 449–480, 2010.
- [25] R. Pichler, *Agile Product Management with Scrum*. Addison-Wesley, 2010.
- [26] **A.-P. Avasiloaie**, A. Semenescu, E.-C. Popovici, and I.-C. Chiva, "Enhancing agile requirement discovery with AI-driven multimodal systems and synthetic user story generation," *Scientific Bulletin Series C: Electrical Engineering and Computer Science*, ISSN 2286 - 3540
- [27] **A.-P. Avasiloaie**, A. Semenescu, E.-C. Popovici, R. Craciunescu, I.-C. Chiva, "Leveraging NVIDIA AI technologies in the development of REQAPP: a machine learning platform for gathering and defining requirements for smart cities applications," 2025. "Smart Cities International Conference (SCIC) Proceedings," V12, (Sep. 2025), pp. 465–480.
- [28] **A.-P. Avasiloaie**, A. Semenescu, E.-C. Popovici, R. Craciunescu, I.-C. Chiva, "AI-driven decision support for SCRUM team selection in smart city software development projects," 2025. "Smart Cities International Conference (SCIC) Proceedings," V12, (Sep. 2025), pp. 453–464.
- [29] I. Altawaiha and A. Al-Hgaish, "ClassDiagGen Tool: Fine-Tuning the GPT-3 Model for Automated Class Diagram Generation from Textual Descriptions," *Research Square*, 2024. [Online]. Available: <https://www.researchsquare.com/> [Accessed: Aug. 12, 2025].
- [30] O. Buruk, "Academic Writing with GPT-3.5: Reflections on Practices, Efficacy and Transparency," *arXiv preprint*, 2023. doi: 10.48550/arXiv.2304.
- [31] T. -C. Ureche, E.-C. Popovici, S. Halunga, **A.-P. Avasiloaie**, L. Boicescu, D. Țurcanu, "Architecting Secure Smart Infrastructures with AI Microservices and Autonomous Agents: A State-of-the-Art Review and Healthcare Use Case," 2025. "IEEE International Black Sea Conference on Communications and Networking," 23–26 June 2025. Chisinau, Moldova
- [32] T.-C. Ureche, E.-C. Popovici, S. Halunga, **A.-P. Avasiloaie**, L. Boicescu, D. Țurcanu, "Architecting Secure Smart Infrastructures with AI Microservices and Autonomous Agents: A State-of-the-Art Review and Healthcare Use Case," 2025. "IEEE International Black Sea Conference on Communications and Networking," 23–26 June 2025, Chisinau, Moldova.
- [33] A. Haile, *AI-Driven Software Testing Automation: Machine Learning Strategies for Performance Optimization in Distributed Networks*, 2024.

- [34] A. Haile, Innovating Software Testing with AI and Machine Learning: Automation for Efficient Distributed Network Management, 2024.
- [35] S. Mansour, “Atlassian welcomes AI to the team,” Atlassian Blog, 2023. [Online]. Available: <https://www.atlassian.com/blog/announcements/atlassian-intelligence-ga> [Accessed: Aug. 12, 2025].
- [36] T. Pîrcălabu, N. Țăpuș, and A. I. Damian, “Programming assistance based on Neural AI OS Platform,” Revista Română de Informatică și Automatică, p. 43, 2024. doi: 10.33436/v34i4y202404.
- [37] S. M. T. H. Rimon, Leveraging Artificial Intelligence in Business Analytics For Informed Strategic Decision-Making: Enhancing Operational Efficiency, Market Insights, And Competitive Advantage, 2024.
- [38] M. Stephen, The Intersection of Technology and Writing Support: How ChatGPT is Changing Writing Center Dynamics, 2024.
- [39] S. Zhang et al., “Empowering Agile-Based Generative Software Development through Human-AI Teamwork,” arXiv preprint, 2024. doi: 10.48550/arXiv.2407.15568.