Universitatea Politehnica din București
Facultatea de Automatică si Calculatoare
Departamentul de Automatică și Informatică Industrială

Rezumat Teză de Doctorat

Deployment Automat al Instrumentelor de Modelare Furnizate ca Servicii in Cloud

Prezentat de

Drd.Ing. Marian LĂCĂTUȘU

Supervizat de

Prof.Dr.Ing. Anca Daniela IONIŢĂ

2023

București, Romania

Abstract

Trecerea la cloud este un subiect care tinde să fie prezent în toate întreprinderile care și-au digitalizat activitățile. Avantaje precum recuperarea în caz de dezastru, colaborarea și economiile de costuri au atras multe organizații să adopte o astfel de abordare orientată spre servicii, iar unele dintre ele și-au mutat întreaga activitate într-un cloud public. De asemenea, în contextul pandemiei de COVID-19, necesitatea colaborării de la distanță între profesori și grupuri de elevi a devenit un aspect educațional important. Dificultățile au venit din faptul că marea majoritate a platformelor de e-learning existente nu erau suficient de pregătite pentru scalare, sau profesorii nu erau pregătiți să folosească exclusiv acest tip de predare. Aceste limitări s-au simțit și în activitățile de cercetare, unde deficitul de colaborare între cercetători a devenit o problemă importantă în condiții de izolare. Scopul meu a fost să rezolv unele dintre limitările prezentate și să ajut studenții, profesorii și cercetătorii în activități de cercetare și educație privind modelarea orientată pe obiecte. Drept urmare, propun aici o platformă multicloud în care studenții, profesorii și cercetătorii care se bazează pe colaborare își pot implementa cu ușurință mediul software pentru a lucra de la distanță. Sunt puțini ingineri specializați atât în modelare, cât și în cloud computing la un nivel tehnic înalt. Această abordare dorește să creeze metode și instrumente care să fie utilizate de inginerii specializați doar într-unul dintre domenii pentru a putea îndeplini cu succes astfel de sarcini interdisciplinare. Un aspect foarte important este complexitatea proceselor care trebuie urmate pentru deploymentul automat și manual. Ca urmare, modelez deploymentul manual și automat cu un limbaj standard de modelare a proceselor de afaceri, investighez metricile de complexitate, fac o selecție care se potrivește scopului nostru și le aplic pentru cele două modele de proces. În plus, propun o nouă metrică de complexitate relevantă pentru acest domeniu, evaluez procesele și demonstrez că într-adevăr complexitatea poate fi de până la 3 ori mai mare pentru deploymentul manual. Acest rezultat demonstrează că automatizarea este foarte importantă pentru ca serviciul să fie mai accesibil utilizatorilor, indiferent de cunoștințele tehnice ale acestora. Prin urmare, am conceput un arbore de decizie pentru a sprijini deploymentul automat multicloud și spre a o dezvolta pentru a selecta mediile de modelare și următorii furnizori de cloud: Amazon Web Services, Microsoft Azure și IBM Cloud. Validez soluția pentru deploymentul în cloud a unui mediu de modelare și metamodelare, adică WebGME, cu un exemplu de limbaj de modelare pentru rețele de senzori și un interpretor de modele specific. De asemenea, evaluez soluția din perspectiva monitorizării cloud și performanța serviciului este testată folosind două soluții de monitorizare. Serviciul funcționează bine pe o implementare minimală Kubernetes, iar pachetul Grafana-Prometheus reprezintă alegerea potrivită pentru monitorizarea în cloud. Mai mult, evaluez algoritmul de decizie din perspectiva celui mai valoros rezultat al acestuia, descriu scenariile privind modul în care această soluție poate fi clasificată (PaaS (Platform as a Service) sau SaaS (Software as a Service)), compar și discut asemănările și diferențele pentru multicloud.

Mulțumiri

În primul rând, vreau să mulțumesc conducătorului meu de doctorat, Profesor Anca Daniela Ioniță pentru îndrumare, sprijin și răbdare pe parcursul studiilor de doctorat. Dumneaei este cel mai bun mentor fără de care nu as fi avut aceste realizări.

Vreau să mulțumesc și doamnei Profesoare Mariana Mocanu, domnului Profesor Florin Anton și doamnei Profesoare Adriana Olteanu. În calitate de membri ai comisiei de doctorat, dumnealor au oferit comentarii si întrebări ce m-au ajutat să-mi îmbunătățesc munca.

De asemenea, vreau să mulțumesc fratelui meu Florin Lăcătușu și bunului meu prieten Ioan Damian pentru colaborarea lor la mai multe articole publicate.

Nu în ultimul rând, vreau să mulțumesc familiei mele pentru sprijinul acordat în acești ani de studii doctorale.

Table of Contents

Listă de Tabele

Listă de Figuri

1. Introducere

1.1 Context și Motivație

Trecerea la cloud este un subiect care tinde să fie prezent în toate întreprinderile care și-au digitalizat activitățile. Aceasta include necesitatea de a lucra cu medii software specifice diverselor domenii de afaceri, accesate ca servicii suportate de diverși furnizori de cloud. În multe domenii de afaceri, oamenii sunt interesați să aibă o soluție bazată pe cloud, care este mai ușor de întreținut în comparație cu implementările on-premise. Avantaje precum recuperarea în caz de dezastru, colaborarea și economiile de costuri au atras multe întreprinderi să adopte o astfel de abordare orientată spre servicii, iar unele dintre ele și-au mutat întreaga activitate într-un cloud public [1]. Se știe că deploymentul în cloud îi scutește pe utilizatori de îndeplinirea sarcinilor obositoare de întreținere și alocare a resurselor [2]. Dincolo de aceasta, pentru a sprijini tot ceea ce este necesar pentru o aplicație, inclusiv infrastructură, date și instrumente de dezvoltare, se poate adopta o soluție Environment as a Service (EaaS). O astfel de soluție este multicloud, ce necesită automatizare pentru implementarea și configurarea mediului și trebuie să gestioneze consumul de resurse. De asemenea, în contextul pandemiei de COVID-19, necesitatea colaborării de la distanță între profesori și grupuri de elevi a devenit un aspect educațional important. O posibilă soluție este mutarea întregii activități în Cloud, oferind astfel și alte avantaje pe lângă colaborarea atât de necesară, precum recuperarea în caz de dezastru și gestionarea resurselor. Deoarece economiile de costuri sunt acum mai importante ca niciodată, trebuie să facem alegerea corectă între furnizorii de cloud unde să implementăm mediile educaționale. O platformă axată pe deployment și colaborare multicloud poate facilita pe cineva să beneficieze de avantajele menționate mai sus, oferind simplitate în utilizare pentru echipele de studenți și cercetători care trebuie să își continue activitatea în perioada de pandemie. Învățarea la distanță este un tip de educație care este considerat orientat spre viitor și avantajează atât elevul, cât și profesorul datorită eliminării constrângerilor precum timpul și spațiul [3]; în special, îi ajută pe studenții care locuiesc într-o locație îndepărtată și nu pot participa la cursuri în persoană. În zilele noastre, tema învățării la distanță a devenit crucială din cauza pandemiei de COVID-19 care a forțat închiderea unor instituții precum școli și universități. Această situație neașteptată a scos în evidență și vulnerabilitățile colaborării la distanță în scopuri de învățare și cercetare, iar multe instituții de învățământ din întreaga lume au fost prinse nepregătite pentru acest tip de criză [4]. Dificultățile au venit din faptul că marea majoritate a platformelor de e-learning existente nu erau suficient de pregătite pentru extindere, sau profesorii nu erau pregătiți să folosească exclusiv acest tip de predare. Aceste limitări s-au simțit și în activitățile de cercetare, unde deficitul de colaborare între cercetători a devenit o problemă importantă în condiții de izolare.

Scopul meu a fost să rezolv unele dintre limitările prezentate, să ajut studenții și profesorii în activități de cercetare și educație privind modelarea orientată pe obiecte. Învățarea la distanță privează profesorii, cercetătorii și studenții de hardware-ul care a fost prezent în laboratoarele școlare sau în unitățile de cercetare; totuși, o parte din acest hardware poate fi înlocuit prin mutarea în întregime în Cloud [5]. Acest lucru poate fi aplicat și instrumentelor de modelare orientate pe obiecte, care pot beneficia de avantaje precum colaborarea și recuperarea în caz de dezastru. Propun aici o platformă în care studenții, profesorii și cercetătorii care se bazează pe colaborare își pot implementa cu ușurință mediul pentru a lucra de la distanță. Cu toate acestea, nevoia de colaborare este esențială pentru utilizatorii care contribuie la același proiect.

Sunt puțini ingineri specializați atât în modelare, cât și în cloud computing la un nivel tehnic înalt. Această abordare dorește să creeze metode și instrumente care să fie utilizate de inginerii specializați doar într-unul dintre domenii pentru a putea îndeplini cu succes astfel de sarcini interdisciplinare. De exemplu, platforma îi ajută pe inginerii specializați în cloud să implementeze un mediu de modelare fără a avea cunoștințe de modelare și, desigur, poate ajuta experții în modelare să facă deployment în cloud fără a avea abilități de cloud. Astfel, principalele obiective ale cercetării sunt următoarele:

- Aplicarea avantajelor deploymentului cloud în domeniul modelării prin dezvoltarea unei platforme care asista și automatizează deploymentul multicloud al unui mediu software
- Evaluarea performanței platformei și complexitatea deploymentului
- Integrarea de funcții secundare, cum ar fi interpretorul de modele, pentru a simplifica și mai mult sarcinile care trebuiau efectuate fără a necesita abilități tehnice.

1.2 Metoda de Cercetare

Un prim studiu a fost realizat pentru a implica studenții de master într-o experiență de cercetare colaborativă pentru dezvoltarea unui limbaj de modelare orientat pe obiecte specific domeniului sistemelor de management al pericolelor [6]. În această fază, colaborarea s-a bazat pe prezentări și discuții în clasă, urmată de lucrul cu o aplicație desktop pentru crearea instrumentelor de modelare – Generic Modeling Environment (GME) [5]. Metoda de cercetare a fost clar definită în prealabil, dar colaborarea a fost orientată în principal către o lucrare conceptuală, în timp ce contribuțiile tehnice au fost individuale. Într-o a doua fază, GME a fost implementat într-un sistem Cloud bazat pe IBM Service Delivery Manager (ISDM), replicând o mașină virtuală separată pentru fiecare student ce participă la cursul Model-Driven Engineering [7]. În consecință, elevii au putut lucra în același mediu în clasă și acasă, iar profesorii au avut acces să monitorizeze progresul. Această metoda de utilizare a unui instrument de modelare este un exemplu de Modeling as a Service, o abordare situată deasupra Software as a Service în organigrama Cloud Service Models [8]. Economisirea costurilor este foarte intâlnită în această perioadă, așa că trecerea la Cloud poate sprijini acest lucru, deoarece resursele sunt menținute atâta timp cât utilizatorul are nevoie de ele.

Totuși, nu toți furnizorii de Cloud au aceleași costuri și servicii care pot completa oferta unei platforme de modelare iar preferințele pot diferi de la un utilizator la altul, așa că ideea de multicloud apare în acest caz cu mai multe alternative. Drept urmare, această lucrare se sprijină pe fazele descrise mai sus și introduce două elemente noi. Pe de o parte, se folosește WebGME (Web-based Generic Modeling Environment) [9] pentru a aborda limitările privind facilitățile de colaborare. Pe de altă parte, deploymentul se face în unul dintre cele 3 cloud-uri acceptate: AWS (Amazon Web Services), Azure, IBM Cloud și serviciul OpenShift Online. De asemenea, pentru a beneficia de avantajele precum disponibilitatea ridicată, flexibilitatea și scalarea oferite de abordarea containerizată, am folosit Kubernetes și Docker în locul mașinilor virtuale. Accentul este pus pe simplitate deoarece în contextul actual utilizatorii vizați trebuie să folosească o mulțime de platforme online, ceea ce le îngreunează foarte mult munca. Chiar dacă o platformă are o utilitate teoretică destul de ridicată, poate fi inaccesibilă multor utilizatori dacă are un nivel foarte ridicat de complexitate. Mai mult, nu toți utilizatorii sunt la același nivel în ceea ce privește abilitățile lor tehnice, așa că acest argument trebuie luat în considerare atunci când alegeți o platformă în scop educațional.

La inițierea utilizării resurselor cloud pot exista procese de urmat care au multe sarcini manuale și necesită cunoștințe de specialitate pentru a fi executate. Acest lucru împiedică adoptarea mult mai largă a acestui tip de furnizare de servicii. De exemplu, complexitatea fluxurilor de lucru de calcul de înaltă performanță (HPC) este analizată în [10]. Li şi colab. modelează si analizează fluxurile de lucru pentru planificatorul de scalare automată, managementul joburilor și managementul contorizării și propune o platformă cloud containerizată pentru a le reduce complexitatea și pentru a-și asista utilizatorii și administratorii. Implementarea unei astfel de platforme rămâne o sarcină dificilă. În afară de complexitate, este de asemenea important să se monitorizeze cu atenție performanța unor astfel de servicii pentru mașinile virtuale și containere, pe baza benchmark-urilor [11]. Bystrov şi colab. evaluează serviciile intensive de comunicare și calcul pe resursele cloud virtuale, a căror performanță depinde de comunicarea MPI (Message Passing Interface), maparea încărcării și supraîncărcarea [12].

Ca urmare, m-am concentrat și pe validarea performanței deploymentului automat în comparație cu deploymentul manual, din perspectiva interacțiunii cu utilizatorul și a cerințelor preliminare necesare pentru sarcinile manuale. Folosesc metricile de complexitate existente și propun, de asemenea, o modalitate specifică de a evalua complexitatea sarcinilor umane pentru deploymentul in cloud. Întrebarea este cât de mari sunt diferențele de complexitate între deploymentul manual și automat, astfel încât să motiveze dezvoltarea unei platforme care să automatizeze majoritatea sarcinilor. Acest lucru ar permite unui specialist în domeniul aplicațiilor să adopte abordarea EaaS chiar și fără cunoștințe și know-how specifice în cloud computing. Cu toate acestea, performanța serviciului rezultat este, de asemenea, importantă pentru adoptarea unei astfel de soluții, iar monitorizarea CPU și a consumului de memorie este, de asemenea, necesară.

1.3 Prezentarea Generală a Tezei

Capitolul 2 va prezenta analiza stadiului actual, inclusiv metodele analitice, și limitările în contrast cu obiectivul tezei. Acesta va reflecta domeniile tezei și va prezenta lecțiile învățate din bibliografia studiată. Principalele domenii de cercetare au fost modelare și cloud. Ele sunt, de asemenea, împărțite în subiecte precum metrici, elemente de teorie, implementări și, de asemenea, sisteme precum sistemele de învățare – domeniul principal în care se poate aplica platforma multicloud. Pe lângă cele prezentate anterior, acest capitol va prezenta articole care se referă la metode de implementare și analiza performanței cloud.

Capitolul 3 prezintă obiectivele tezei: cele generale și cele specifice. Aceste obiective se referă la scopurile cercetării și prezintă ceea ce s-a intenționat a fi realizat în anii de studii doctorale. Alături de descrierea obiectivelor, capitolul 3 prezintă și metoda de cercetare care a fost utilizată pentru atingerea obiectivelor. Ca urmare, fiecare pas care a fost făcut este descris pornind de la stadiul actual aferent până la cadrele utilizate pentru dezvoltarea și arhitectura aplicației. Aceasta reprezintă fluxul cercetării.

Capitolul 4 prezintă în detaliu contribuțiile personale realizate pentru această teză. Descrie ce s-a făcut pentru atingerea obiectivelor prezentate în Capitolul 3. Începând cu descrierea deploymentului manual și automat, și descrierea soluției pentru deploymentul automat, metodele de deployment sunt validate prin utilizarea metricilor de complexitate a procesului de afaceri și prin propunerea unei metrici de complexitate nouă și originală. După ce este prezentată complexitatea proceselor de afaceri, sunt explicate mijloacele de realizare a metricilor cloud, pentru a obține rezultatele complexității pentru deploymentul cloud. Capitolul

continuă cu algoritmul de decizie care a fost creat pentru a stabili detaliile de deployment și cele cinci rezultate care rezultă din acesta. Capitolul se încheie cu o prezentare generală a tehnologiilor utilizate pentru implementarea platformei.

Capitolul 5 reprezintă verificarea și validarea contribuțiilor expuse în capitolele precedente. Începe cu migrarea unui metamodel GME la un mediu WebGME implementat pe clusterul Kubernetes din IBM Cloud. Aceasta a validat rolul principal și funcționalitatea platformei – funcționează conform așteptărilor și, de asemenea, identifică dificultățile migrării metamodelului de la GME la WebGME. Capitolul continuă cu rezultatele de performanță ale deploymentului cloud reprezentate de rezultatele monitorizării, inclusiv evaluarea algoritmului de decizie prezentat în Capitolul 4. Capitolul persistă cu o discuție care vizează mediul de modelare în contextul unei arhitecturi de microservicii construite pe Docker și Kubernetes. Cu toate acestea, rezultatele metricilor de complexitate sunt analizate împreună cu rezultatele performanței deploymentului Cloud. Capitolul se încheie cu o discuție despre platforma de deployment automat și capacitatea principală pe care o oferă.

Capitolul 6 prezintă concluziile alături de o prezentare generală a contribuțiilor originale și a perspectivelor viitoare. Include și lista articolelor publicate.

Anexa 1 prezintă în detaliu platforma de deployment multicloud din perspectiva backend și frontend. O parte a API-ului furnizată de backend este prezentată în detaliu alături de specificația Swagger. Frontend-ul Angular este prezentat alături de caracteristicile oferite de platformă. Acesta reprezintă o imagine de ansamblu a ceea ce a fost realizat practic pentru a susține teoria propusă. În cele din urmă, am prezentat vizualizarea monitorizării implementabile dintr-un CLI conectat la cluster-ul Kubernetes.

2. Analiza Stadiului Actual

Cele două domenii principale de cercetare pentru teză sunt modelarea și deploymentul în cloud. Cele două subiecte principale sunt, de asemenea, împărțite în secțiuni care vizează elementele teoretice, performanța și validarea. Cercetarea privind sistemele de învățare este tratată într-un subcapitol separat.

2.1 Aspecte de Modelare

Am studiat domenii precum Model și Modeling as a Service, Model-Driven Architecture și Model-Driven Engineering. Din articolele din categoria Model și Modeling as a Service, am dobândit cunoștințele necesare legate de implementarea unui model și medii de metamodelare în cloud și modul de lucru cu astfel de instrumente. În plus, am dobândit cunoștințe teoretice legate de acestea și deploymentul unui mediu similar în cloud. Multe dintre articolele prezente au avut o scurtă descriere a infrastructurii care a furnizat modelul/modelarea. În plus, am aflat despre diferențele dintre instrumentele de modelare precum GME, WebGME, AToMPM, MDEForge etc. Cele mai multe dintre aceste instrumente aveau caracteristici comune, cum ar fi colaborarea, scalabilitatea și persistența modelului. Aceste caracteristici au fost comparate din perspectiva performanței. Drept urmare, mi-am făcut o idee legată de aceste instrumente și de capacitățile lor. Un alt subiect studiat este arhitectura Model-Driven. Am aflat despre istoria și teoria MDA alături de unele analize ale abordărilor culminate cu exemple practice. În general, această abordare încurajează dezvoltarea mai rapidă a software-ului împreună cu

flexibilitatea tehnologiilor utilizate. Următorul domeniu studiat este legat de modelarea multi-paradigmă în care, pe parcursul articolelor studiate, am aflat câteva aspecte teoretice legate de model, metamodel, semantică alături de câteva exemple oferite de autori. Alte studii au fost legate de abordarea aplicată în domeniul sistemelor ciber-fizice și de descrierea unei astfel de implementări.

2.2 Aspecte de Cloud Computing

Cloud computing este al doilea domeniu de studiu al tezei. Această secțiune descrie lucrări conexe care se referă la Kubernetes, deployment în cloud, metrici de complexitate pentru deployment în cloud și analiza performanței deploymentului.

3. Deployment Multicloud al Instrumentelor de Modelare

3.1 Obiective

Obiectivul principal al cercetării este aplicarea avantajelor deploymentului cloud în domeniul modelării, oferind o punte între cele două seturi de abilități: modelare și cloud computing. Ca urmare, experții în cloud vor putea implementa un mediu de modelare și interpretoare de modele fără a avea abilități de modelare, iar experții în modelare vor putea implementa instrumente de modelare în cloud fără a fi nevoie de abilități de cloud. Prin urmare, un obiectiv specific este de a oferi o platformă multicloud care poate implementa medii de modelare alături de interpretoare de modele pe unul dintre cele trei cloud-uri: AWS, Azure, IBM Cloud și serviciul OpenShift Online. Capacitățile platformei pot fi extinse, făcându-l un hub în care utilizatorii pot implementa mediul de modelare pe care l-au ales și pot controla deploymentul așa cum li se pare potrivit. Deploymentul mediilor se face în câțiva pași de către utilizatorul înregistrat și poate fi utilizat și de utilizatorii non-tehnici. Platforma poate implementa mediul de modelare și interpretoare de model într-unul dintre cele trei cloud-uri și, de asemenea, OpenShift Online dacă avantajele precum monitorizarea integrată sunt necesare pentru a fi exploatate de către utilizatori. În plus, platforma are capacitatea de a ajuta utilizatorii să aleagă instrumentul de modelare potrivit , punându-le un set de întrebări. Pe baza răspunsurilor, algoritmul va sfătui utilizatorii pe ce platformă să implementeze în Cloud. Datorită acestui sfat, decizia va fi mai simplă și, de asemenea, va diferenția utilizarea și utilitatea ambelor medii de modelare disponibile.

În contextul pandemiei de coronavirus, această platformă ar putea ajuta, de asemenea, studenții și cercetătorii în sarcinile lor care sunt desfășurate în condiții de izolare impusă. De exemplu, această platformă poate completa un mediu de laborator și poate oferi profesorilor o modalitate mai controlabilă pentru resursele desfășurate. În plus, studenții vor putea colabora la proiecte profitând de caracteristicile oferite de platformă și de avantajele deploymentului cloud. Acest mediu poate fi distrus după fiecare utilizare pentru a economisi bani. Cu abonamentul de tipul „pay-as-you-go” ce este oferit de majoritatea furnizorilor de cloud, costurile pot fi reduse pentru mediile care nu au scopul de a rula continuu pentru o perioadă lungă de timp. Ca urmare, resursele pot fi șterse, iar modelele/proiectele pot fi salvate pe stocarea în cloud care va rămâne pe abonamentul utilizatorului. Acest tip de stocare nu este la fel de costisitor ca un cluster Kubernetes sau chiar ca o mașină virtuală de dimensiuni medii. De asemenea, în același context, profesorii și studenții întâmpină unele dificultăți în adoptarea noilor tehnologii, deoarece din cauza lockdown-ului impus au apărut multe platforme care să ofere mai multe

beneficii acestui tip de utilizator. Cele mai multe dintre aceste platforme pot fi greu de utilizat sau pot apărea dificultăți din cauza multitudinii de platforme noi deci; în opinia mea, simplitatea este cheia și punctul central atunci când se începe dezvoltarea unei noi platforme. Ca urmare, principalele obiective sunt:

1. Propunerea unei noi platforme multicloud care poate implementa automat un mediu de modelare într-un cloud la alegerea utilizatorului.
2. Evaluarea metodelor de deployment automat și manual și compararea acestora pe baza unor criterii definite de metrici de complexitate.
3. Analiza performanței deploymentului cloud folosind metrici cloud.
4. Elaborarea și implementarea unui nou algoritm de decizie pentru alegerea detaliilor de deployment în cloud.
5. Furnizarea de exemple de integrare a interpretoarelor de modele cu platforma multicloud și evaluarea rezultatelor.

3.2 Metodă

Pe baza obiectivelor prezentate anterior, metoda de mai jos determină activitățile realizate pentru atingerea acestor obiective.

Primul și cel mai important lucru de înțeles este cum se poate realiza ceea ce se propune, pentru a determina modul în care alți cercetători au abordat această problemă și ce instrumente sunt cele mai potrivite pentru o astfel de implementare. Fig. 3.1 descrie etapele care au fost făcute pentru cercetare, realizarea soluției și obținerea rezultatelor. Toți pașii prezentați se bazează pe cei anteriori.

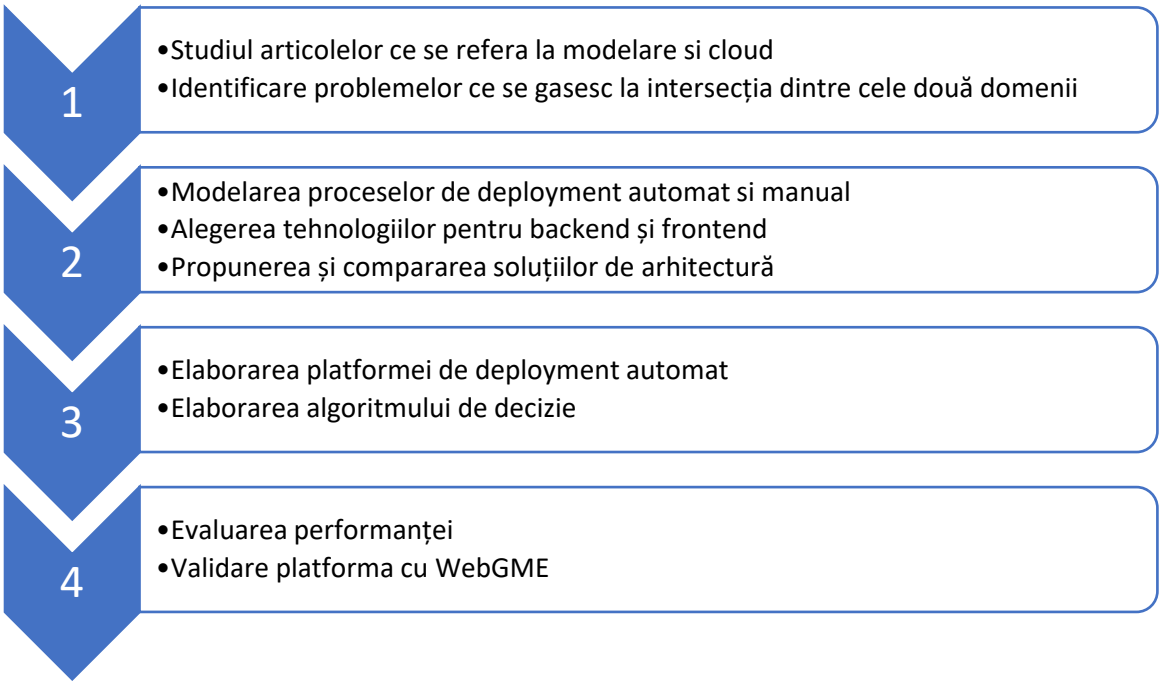

Fig. 3.1 Metoda de implementare a soluției

4. Contribuții la Deploymentul Automat al Instrumentelor de Modelare

4.1 Deploymentul Manual al unui Instrument de Modelare. Exemplu pentru IBM Cloud

Primul test pentru deploymentul manual a fost implementarea instrumentului de modelare ales - WebGME în IBM Cloud. Acest lucru este foarte important pentru utilizatorii care doresc să contribuie la un proiect privat de modelare și poate fi realizat folosind containerizarea (cu tehnologii precum Docker și Kubernetes). Înainte de deployment, este necesar să fie implementat un serviciu Kubernetes, folosind licența academică IBM Cloud. Există un pod care gestionează cel puțin un container; în scopuri de containerizare, a fost folosit Docker, bazat pe o imagine personalizată WebGME creată cu un Dockerfile. Fișierul Docker pentru WebGME necesită un fișier de configurare care indică containerul Mongo. Deci, aceasta este prima cerință prealabilă necesară înainte de a construi imaginea.

Deploymentul manual şi cerinţele necesare pentru ca implementarea să fie finalizată sunt prezentate în Fig. 4.1.

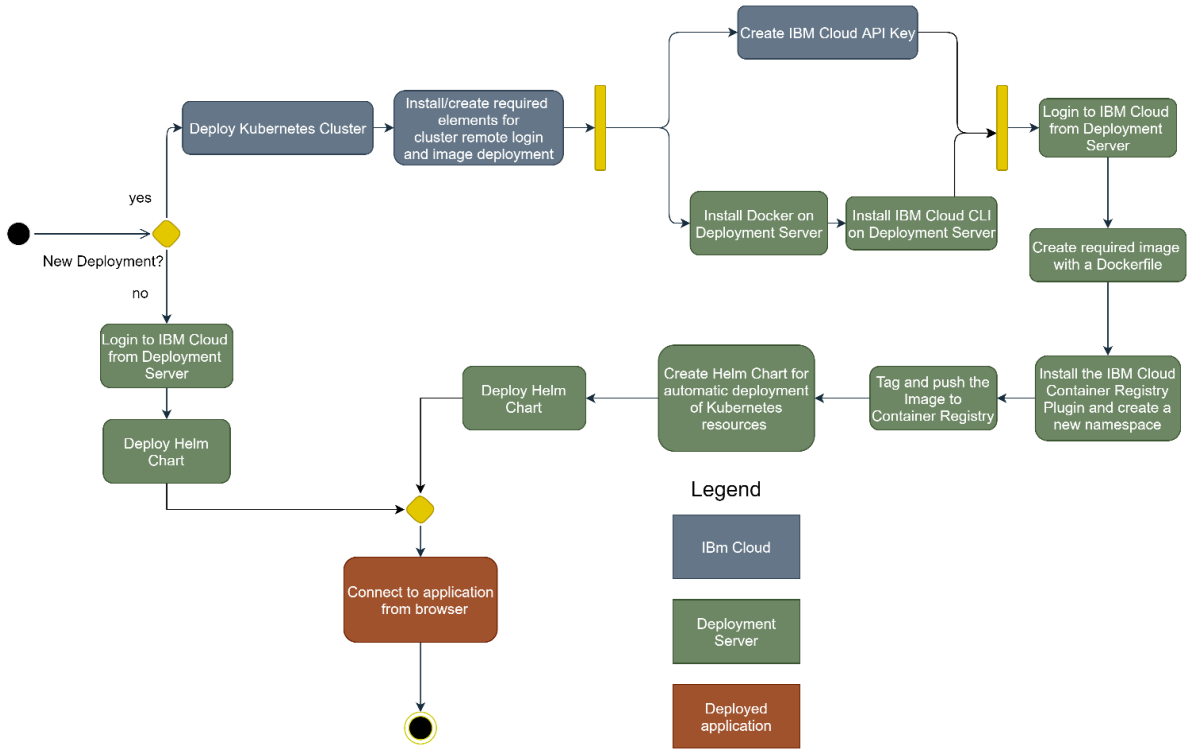

Fig. 4.1 Sarcini ce trebuie executate pentru deploymentul manual al instrumentului de modelare (IBM Cloud)

Fig. 4.1 descrie procesul de deployment al instrumentului de modelare în IBM Cloud. Această metodă necesită cunoștințe tehnice avansate din partea utilizatorului și este, de asemenea, foarte costisitoare din perspectiva timpului. Procesul începe cu implementarea clusterului Kubernetes dacă este un nou deployment și continuă cu cerințele preliminare pentru un server de deployment. Cerințele prealabile implică crearea cheii API (pe platforma IBM Cloud) și instalarea Docker și IBM Cloud CLI pe serverul de deployment. După aceea, utilizatorul trebuie să se conecteze la spațiul de lucru IBM Cloud de pe serverul de deployment. În

continuare, utilizatorul va crea o imagine pentru instrumentul de modelare. Această imagine se bazează pe un Dockerfile care trebuie configurat. Sarcinile de configurare continuă cu instalarea plugin-ului pentru IBM Cloud Container Registry util pentru conectarea la această componentă. De asemenea, trebuie creat un namespace. Namespace-ul are rolul de a grupa resursele Kubernetes implementate. După aceea, imaginea este trimisă în IBM Cloud Container Registry. Următorii pași implică crearea deploymentului și serviciilor pentru WebGME și MongoDB. Aceste servicii vor fi implementate automat pe clusterul Kubernetes, mai degrabă decât să implementeze serviciile unul câte unul. Procesul se termină cu conexiunea la mediul nou implementat. Dacă condițiile suplimentare sunt deja făcute și aceasta nu este un deployment nou, problema va fi mai simplă. Utilizatorul trebuie doar să se conecteze la contul IBM Cloud și să implementeze Helm Charts care sunt deja create. Cu ajutorul Helm Charts, utilizatorul poate distruge mediul după bunul plac. Acest lucru face ca acțiunea de deployment/distrugere să fie foarte accesibilă. Aceste acțiuni necesită multe cunoștințe tehnice și know-how pentru aceste tehnologii. În opinia mea, un utilizator nu ar trebui să aibă nevoie de acest tip de abilitate doar pentru a implementa un mediu mai eficient.

4.2 Deploymentul Automat din Platforma de Multicloud

Utilizarea platformei multicloud simplifică munca utilizatorului care trebuie făcută. În acest caz, utilizatorul va trebui în continuare să implementeze cluster-ul Kubernetes. Acest lucru se poate face foarte ușor din consola web IBM Cloud. De asemenea, utilizatorul trebuie să creeze și să recupereze o cheie API. Această sarcină diferă de la un furnizor de cloud la altul deoarece conexiunea la clusterul Kubernetes necesită alte mijloace care sunt specifice unui furnizor de cloud. În cazul IBM Cloud, este nevoie să fie furnizată de către utilizator doar o cheie API la pasul specific. În primul rând, utilizatorul trebuie să se înregistreze pe platformă după ce cluster-ul Kubernetes este implementat și cheia API este preluată. Apoi, utilizatorul ar trebui să creeze un proiect pentru a avea posibilitatea de a adăuga utilizatori și conexiuni. Acest utilizator poate avea mai multe proiecte, astfel încât proiectul curent care va fi în uz trebuie setat ca activ. După aceea, utilizatorul va selecta la ce cloud dorește să se conecteze. Opțiunile actuale sunt IBM Cloud, AWS, Azure și OpenShift Online. După aceste selecții, utilizatorului i se va solicita să adauge cerințele de conectare. Pentru acest exemplu, utilizatorului i se va cere să furnizeze cheia API preluată la pasul anterior și numele clusterului Kubernetes. După aceea, utilizatorul va seta conexiunea activă, deoarece, ca și în cazul proiectului, un utilizator poate avea mai multe conexiuni la mai multe clustere Kubernetes care se gasesc în cele trei cloud-uri și OpenShift Online acceptate. Pasul final este reprezentat de deploymentul aplicației și de conectarea utilizatorului la aceasta.

În comparație cu pașii pe care îi întreprinde un utilizator cu metoda prezentată anterior, în acest caz, cunoștințele tehnice avansate care sunt vitale pentru partea de configurare nu sunt necesare și se fac automat din platforma. Practic, utilizatorul va interacționa doar cu interfața prietenoasă a furnizorului de cloud și platforma multicloud. Serverul de deployment este practic inexistent pentru utilizator și toate sarcinile sunt gestionate de backend-ul aplicației. Sarcinile descrise sunt reprezentate în Fig. 4.2 de mai jos.

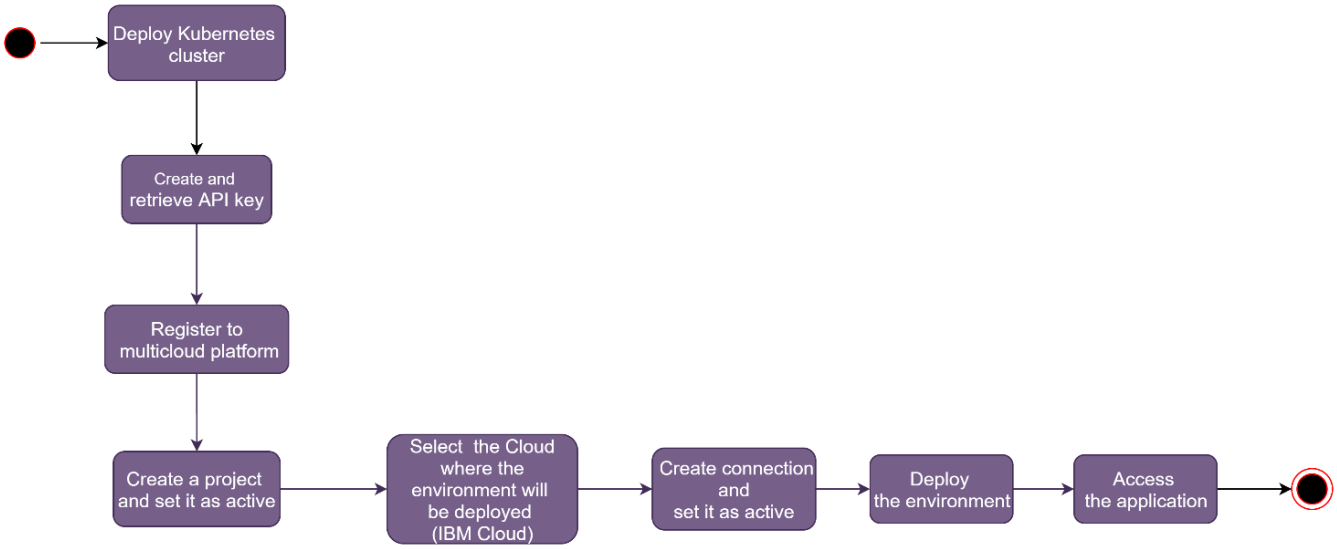

Fig. 4.2 Deploymentul instrumentului de modelare din platforma multicloud

4.2.1 Deploymentul automat al interpretorului de modele

Au fost propuse două metode pentru a oferi interpretorul de modele ca serviciu. Prima soluție a implicat utilizarea stocării în cloud, aceasta este o problemă dacă luăm în considerare soluția de ansamblu, deoarece utilizatorul este obligat să investească în metode costisitoare pentru a putea accesa această caracteristică. Ca urmare, am ales a doua metodă care implică utilizarea unor interpretoare încapsulate în imaginea docker. Dezavantajul acestei metode este utilizarea mai multor imagini. Dezavantajul este că este greu de gestionat și actualizat mai multe imagini. De exemplu, dacă este lansată o nouă versiune de WebGME, toate imaginile trebuie actualizate în consecință. În comparație cu prima metodă, această problemă este acceptabilă deoarece este de partea dezvoltatorului, nu de partea utilizatorului. Dezvoltatorul trebuie să își actualizeze imaginile și să le furnizeze utilizatorului. Prima metodă implică faptul că utilizatorul trebuia să achiziționeze un serviciu în cloud (stocare). Acest lucru este inacceptabil în contrast cu această soluție.

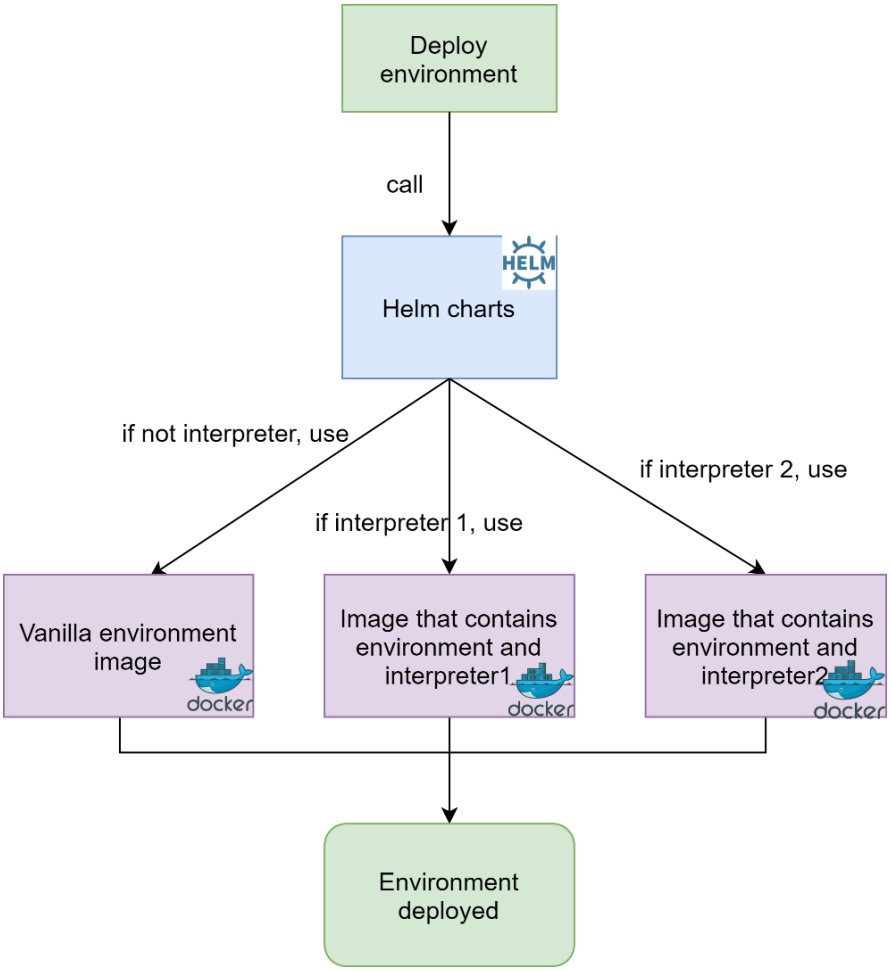

Fig. 4.3 Utilizare mai multor imagini Docker pentru deploymentul cu Helm Charts

Fig. 4.3 ilustrează metoda imaginilor docker multiple din perspectiva deploymentului cu Helm Charts. La prima vedere, acest lucru va avea același efect asupra utilizatorului: mediul va fi implementat în același mod. În funcție de interpretorul selectat, mediul va fi implementat cu acel plugin inclus și va fi afișat în consola web WebGME, sub meniul de selecție al plugin-ului.

4.3 Metrici de Complexitate pentru Deploymentul Manual și Automat

După prezentarea soluției în subcapitolele anterioare, voi prezenta metricile de complexitate aplicate pentru a determina care este diferența de complexitate între deploymentul manual prezentat în Fig. 4.1 și deploymentul automat prezentat în Fig. 4.2. Procesele de deployment manual și automat sunt analizate în Capitolul 4.3.1, mai întâi pe baza metricilor de complexitate prezentate în literatura științifică și apoi cu complexitatea sarcinilor umane, o metrică specifică definită pentru domeniul de aplicare al lucrării noastre.

4.3.1 Alegerea metricilor de complexitate pentru analiza procesului de deployment și rezultate

În primul rând, folosesc o metrică clasică, complexitate ciclomatică [13], pentru calcularea complexității proceselor de deployment. Complexitatea ciclomatică, introdusă de Thomas McCabe în 1976, determină complexitatea programului.

Prin urmare, complexitatea ciclomatică este 3 pentru procesul de deployment manual și 1 pentru cel automat. O valoare mai mică a complexității pentru cel din urmă indică faptul că deploymentul automat necesită mai puțină muncă din partea utilizatorului și confirmă necesitatea unei platforme dezvoltate în acest scop.

În al doilea rând, aplic formula complexității Yaqin, bazată pe o serie de metrici care sunt calculate pentru fiecare componentă a procesului[14].
După calcularea tuturor ecuațiilor, complexitatea Yaqin (YC) este 78,5 pentru deploymentul manual și 35 pentru deploymentul automat.

În al treilea rând, în scopul analizei mele, definesc o metrică specifică bazată pe cunoștințele tehnice necesare pentru a finaliza o sarcină manuală. În acest scop, fiecărei sarcini de proces i se atribuie o pondere, conform următoarelor criterii:

- 1 - sarcină ușoară - este necesar urmărirea unui tutorial
- 2 - sarcină medie – cunoștințe tehnice specifice necesare
- 3 - sarcină grea – cunoștințe tehnice specifice și experiență necesară

Complexitatea procesului din punctul de vedere al utilizatorului care este responsabil pentru executarea sarcinilor manuale este calculată astfel:

$$CH = \sum_{t=1}^{n} Wt \quad (1)$$

unde CH este complexitatea sarcinilor umane implicate în procesul de implementare, Wt este ponderea pentru o anumită sarcină și n reprezintă numărul sarcinii finale.

Pe baza criteriilor, complexitatea rezultată a sarcinilor umane (CH) — calculată cu metrica specifică propusă este de 20 pentru deploymentul manuală și 8 pentru cel automat.

După calcularea complexităților (pentru deploymentul manual și automat) după criteriile explicate, rezultatele sunt cele așteptate. Deploymentul manual este de 2 până la 3 ori mai complex și necesită mai multe cunoștințe tehnice din partea utilizatorului în comparație cu cel automat. Luând în considerare această măsurătoare, deploymentul automat economisește timp și poate fi utilizat de utilizatori fără cerințe tehnice. Așadar, după prezentarea acestor cazuri pot spune în mod corect că deploymentul automat aduce multă valoare întregii experiențe.

4.4 Instrumente de Monitorizare Pentru Analiza Performanței și Metrici Cloud

Alături de metricile de proces studiate în subcapitolul anterior, un subiect la fel de important este legat de metricile cloud și performanța deploymentului pe serviciul Kubernetes. În acest sens, am ales o abordare standard de monitorizare: Prometheus. Conform [15], Prometheus este un instrument de monitorizare, alertare care oferă un limbaj de interogare numit PromQL. Acest limbaj a oferit utilizatorului posibilitatea de a face expresii (interogări) care pot afișa un grafic sau un tabel în timp real. În zilele noastre, Prometheus este foarte popular în lumea containerizării datorită integrării implicite cu platforma Kubernetes - OpenShift. Ca software

de vizualizare pentru Prometheus, am folosit Grafana care se integrează corespunzător cu Prometheus. Prometheus oferă patru tipuri de metrici utilizate în diferite situații [16]:

- Counter - folosit pentru a număra numărul de solicitări, crescând până se reseteaza la zero
- Gauge - un contor care poate merge în sus și în jos; folosit mai ales pentru valorile memoriei și CPU
- Histogram - utilizat pentru a calcula durata cererii și, de asemenea, suma duratei pentru toate cererile
- Summary - asemănător histogramei; în plus, oferă o sumă a tuturor valorilor

4.5 Algoritm de Decizie pentru Stabilirea Detaliilor de Deployment

Acest capitol descrie factorii decisivi pentru deploymentul instrumentelor de modelare din platforma multicloud. Aceste decizii sunt luate prin valorificarea capabilităților arborilor de decizie și, de asemenea, prin furnizarea unui algoritm decizional care ajută un utilizator să decidă detaliile de deployment.

Algoritmul de decizie este aplicat unui set de date dat, care se bazează pe opțiunile utilizatorului. Capitolul 4.5.1 descrie aplicarea algoritmului ID3 pentru a determina condiția de split. La acest pas, nu luăm în considerare cloud-ul în care va fi implementat mediul de modelare.

4.5.1Decizie de Deployment Bazată pe Arborele de Decizie ID3

Aplic algoritmul ID3 la un set de date care reprezintă combinația de decizii de deployment și rezultatele acestora. Primii arbori de decizie ar trebui să prezică în ce condiții ar trebui să fie implementat WebGME. Setul de date a fost creat luând în considerare deciziile de implementare, așa cum sunt prezentate în Tabelul 4.1.

Table 4.1 Decizii de deployment pentru WebGME

Abonament	Limbaj de programare	Colaborare	WebGME
free	C++/Java	yes	yes
paid	C++/Java	yes	yes
fee	javascript	no	yes
paid	javascript	no	yes
free	C++/Java	no	no
paid	C++/Java	no	no

Fig. 4.4 prezintă arborele de decizie rezultat pentru seturile de date prezentate în tabelul 4.1. De asemenea, descrie alegerile pe care un utilizator trebuie să le facă pentru a implementa WebGME.

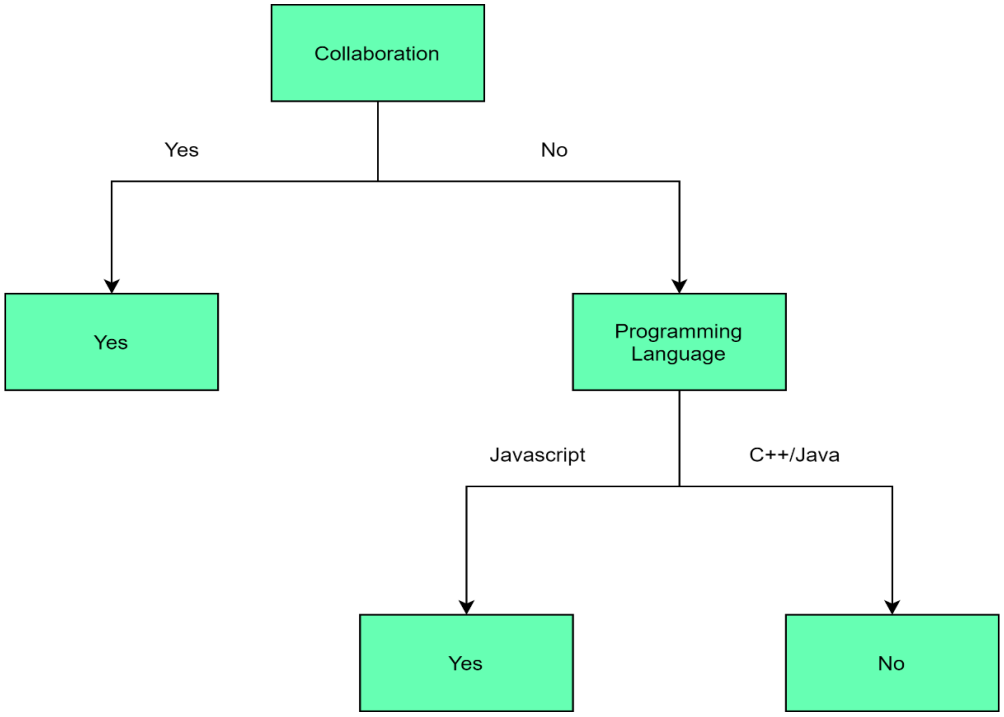

Fig. 4.4 Arborele de decizie construit cu algoritmul ID3

4.5.2 Algoritm de decizie pentru deploymentul GME/WebGME într-un mediu multicloud

După ce am prezis alegerea unui utilizator pe baza preferințelor sale, cum ar fi colaborarea sau limbajul de programare, propunem un algoritm care sfătuiește utilizatorul ce instrument de modelare este mai bine să implementeze și în ce cloud. În comparație cu seturile de date utilizate mai devreme, introduc caracteristici suplimentare pentru a diferenția alegerea de implementare. Sunt luate în considerare următoarele criterii:

- Abonament: gratuit, plătit, rezervat/la cerere
- Limbaj de programare: C++/Java/Javascript
- Colaborare: Da/Nu
- Timp de utilizare: neîntrerupt timp de o lună, mediu distrus după utilizare.

Reprezentarea algoritmului din Fig. 4.5, 4.6 și 4.7 este menită să ajute utilizatorul să decidă ce și unde să implementeze în funcție de întrebare, alegere și rezultat. Pentru secitunea ce implica un abonament plătit, nu ofer sfaturi cu privire la furnizorul de cloud, deoarece utilizatorul trebuie să aibă un abonament activ plătit. Pentru partea gratuită, sfătuiesc alegerea furnizorului de Cloud deoarece un abonament gratuit este accesibil tuturor utilizatorilor.

Primele criterii de split se bazează pe tipul de abonament pe care un utilizator îl poate deține în cloud (Fig. 4.5).

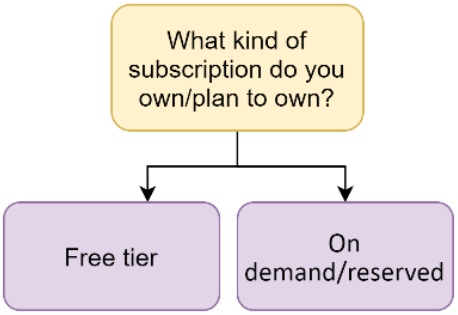

Fig. 4.5 Rădăcina arborelui de decizie

Acest criteriu împarte arborele de decizie în două jumătăți care generează rezultate diferite. Partea ce se refera la abonament gratuit generează trei rezultate, iar partea ce se refera la abonament la cerere/rezervat oferă două rezultate.

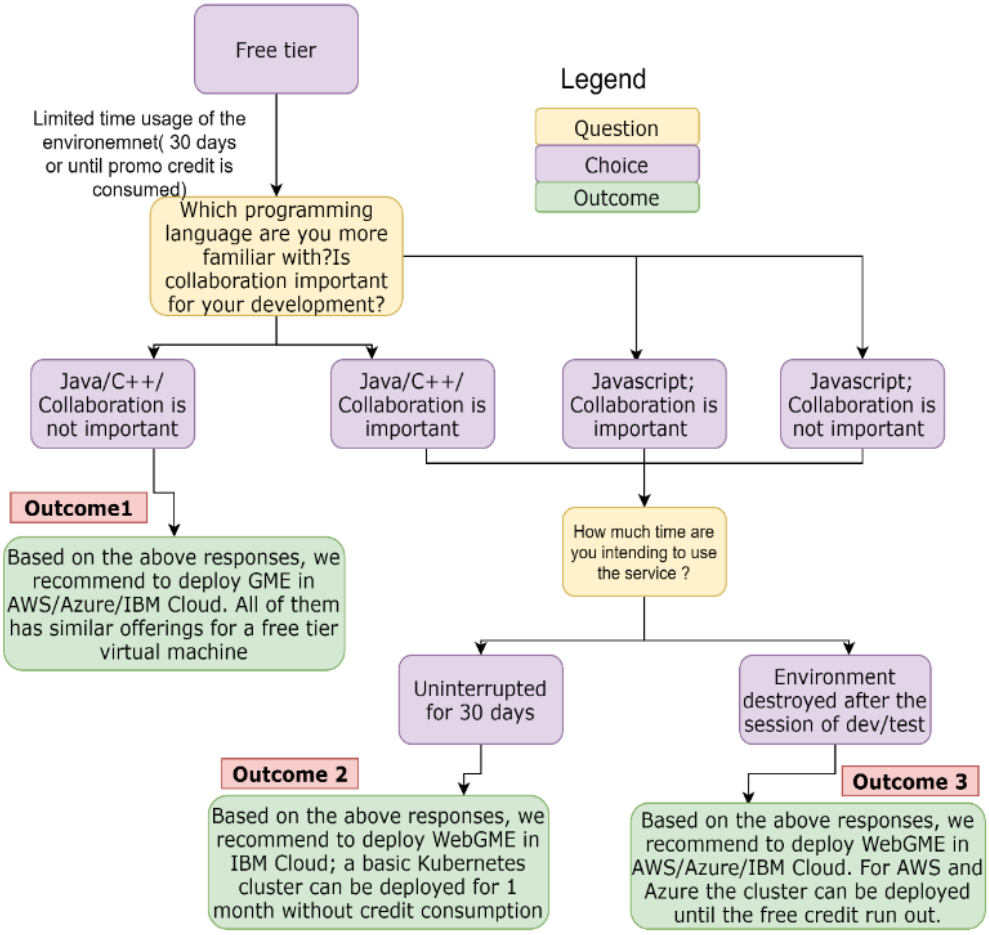

Fig. 4.6 Secțiunea din stânga a arborelui de decizie

Secțiunea din dreapta (Fig. 4.7) a arborelui de decizie se referă la abonamentele plătite și generează două rezultate.

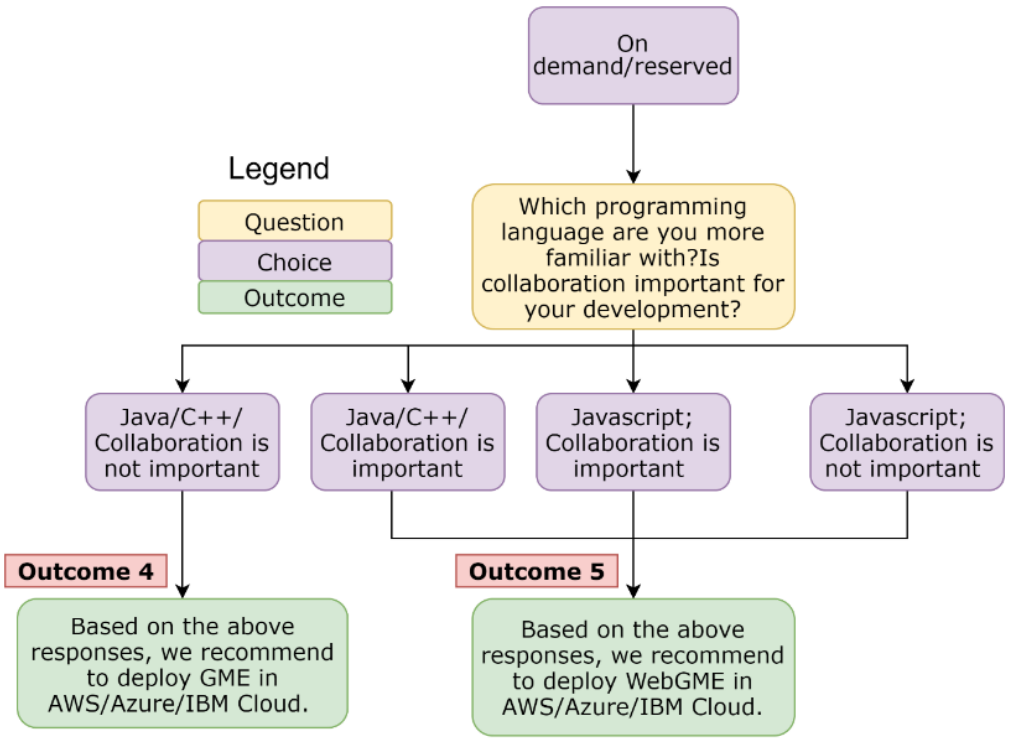

Fig. 4.7 Secțiunea din dreaptă a arborelui de decizie

4.6 Platform Implementation - Technologies

După cum sa menționat anterior, tehnologiile utilizate sunt cele care sunt caracteristice pentru deploymentul Cloud. Aplicația folosește cele mai recente tehnologii de containerizare care sunt disponibile pe furnizorii de cloud aleși: AWS, IBM Cloud, Microsoft Azure și OpenShift Online. Infrastructura conține un server de pe care se realizează implementarea. Acest server trebuie configurat să ruleze Docker, Terraform și Helm. De asemenea, instrumentele CLI pentru furnizorii de cloud trebuie să fie instalate pe serverul de deployment. Scripturile de deployment folosesc aceste tehnologii pentru a implementa mediul de modelare într-una dintre cele trei platforme cloud și OpenShift Online. Alături de tehnologiile de deployment, pe același server, aplicația și serverul web sunt instalate pentru a rula portalul. Platforma multicloud a fost creată folosind Node.js pentru backend și Angular 10 pentru frontend. Aceste tehnologii au fost alese datorită eficienței pe care o poate oferi aplicației pe baza unor criterii precum timpi de încărcare mai rapizi și, de asemenea, utilizarea programării asincrone furnizate de node.js și observabilelor furnizate de framework-ul Angular care este folosit și pentru programarea și mesajele asincrone. care sunt trimise între componentele aplicației. Persistența datelor este asigurată de baza de date MySQL care rulează pe un container Docker pe serverul de deployment care are atașat un volum persistent.

Implementarea în cloud se poate face fără Helm, dar, pentru a avea o implementare complet automată, aceste tehnologii ajută foarte mult la realizarea acestui lucru. Specificația tehnică este prezentată în Fig. 4.8

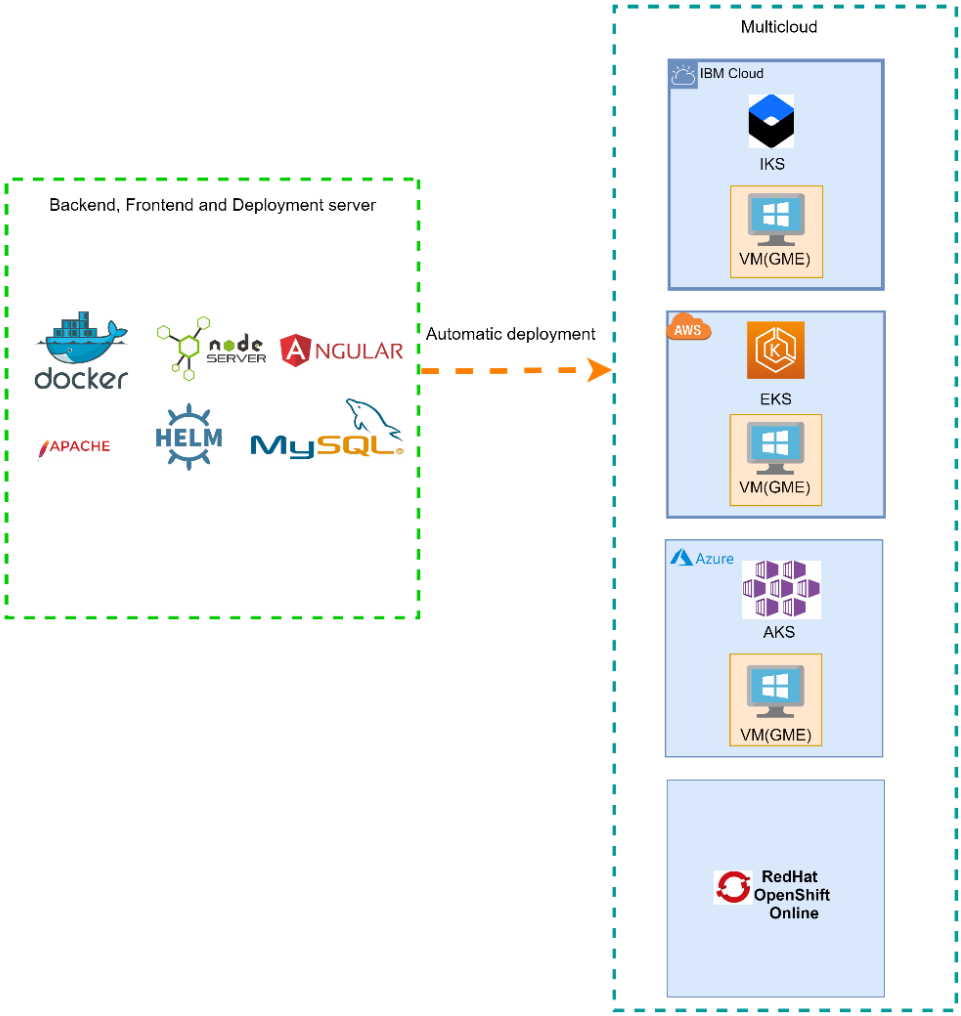

Fig. 4.8 Prezentare generală a tehnologiilor utilizate pentru platformă

În ceea ce privește tehnologiile utilizate pentru implementare, acestea sunt:

- Helm Charts pentru deploymentul resurselor Kubernetes
- Docker pentru crearea și gestionarea imaginilor
- Apache – server web
- MySQL – persistența datelor

5. Evaluare și Validare

5.1 Deploymentul WebGME în cloud - Exemplu de validare a caracteristicilor

Deploymentul WebGME în Cloud a fost testat cu un metamodel care a fost prezentat anterior în [6] și [17]. Fig. 5.1 prezintă o parte din sintaxa abstractă a acestui metamodel, concepută pentru a crea un limbaj de modelare grafică specific rețelelor de senzori, care au devenit importante pentru multe sisteme de colectare a datelor și capabilități de fuziune [18]. Metamodelul a fost reprezentat în GME - mediul de metamodelare la fața locului studiat aici. Un SensorNetwork este compus din mai multe elemente SensorNetworkUnit, care pot fi unități de comunicare, procesare sau detectare – ultimele potenţial compuse din mai mulți senzori. Pot exista, de asemenea, conexiuni între aceste unități, precum și dependențe între unitățile software care fac parte din unitățile de procesare. În plus față de elementele reprezentate în Fig. 5.1, metamodelul conține detalii suplimentare cu privire la aspectele de comunicare, memorie, putere și senzori. Pe baza acestui metamodel, am creat modele în scop de testare, pentru a putea găsi informațiile necesare unei comparații unul lângă altul.

Paradigma creată cu GME a fost importată în WebGME implementat în IBM Cloud, folosind pluginul de importator MetaGME, prin fișierul .xmp. Un astfel de import este caracterizat, totuși, de câteva limitări [19], cum ar fi imposibilitatea de a importa roluri, constrângeri, cardinalitate, aspecte și proprietăți vizuale, precum și utilizarea unor tipuri în loc de roluri. Rezultatul acestui export pentru metamodelul rețelei de senzori este ilustrat în Fig. 5.2.

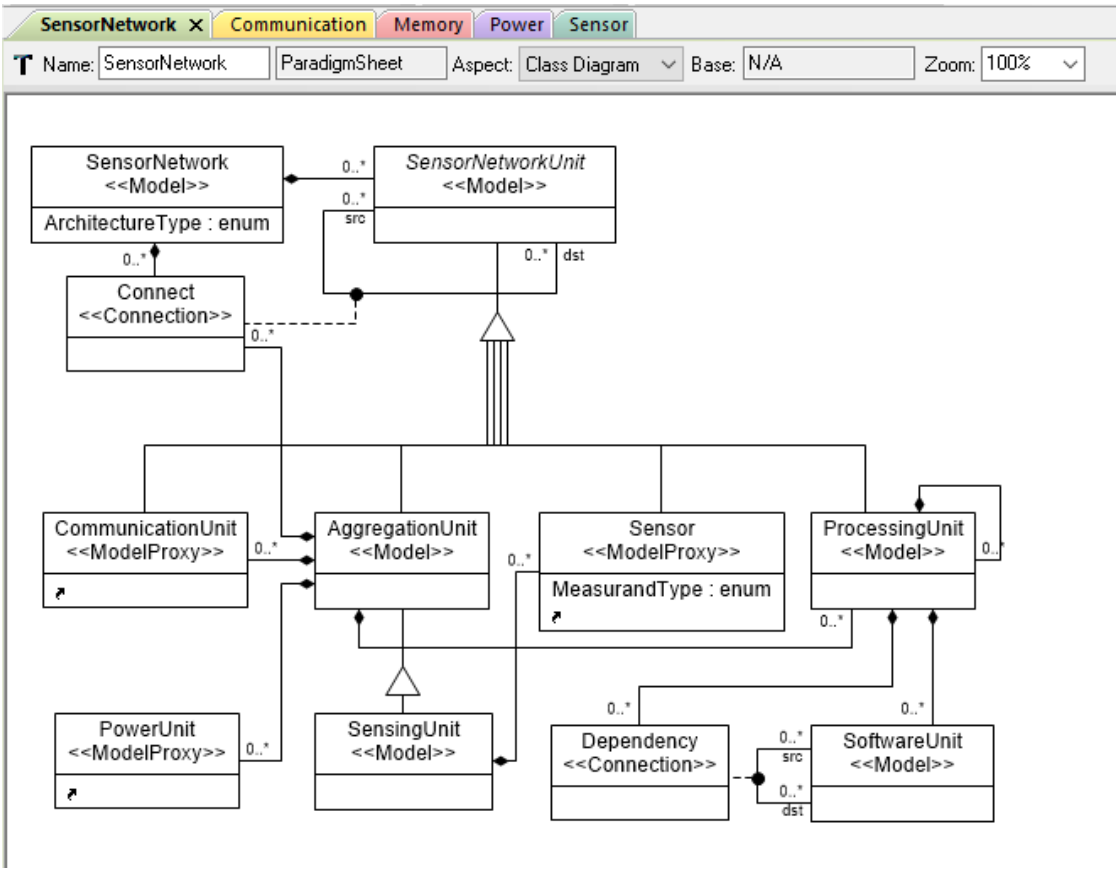

Fig 5.1. Parte din metamodel - instrumentul de metamodelare (dupa [6])

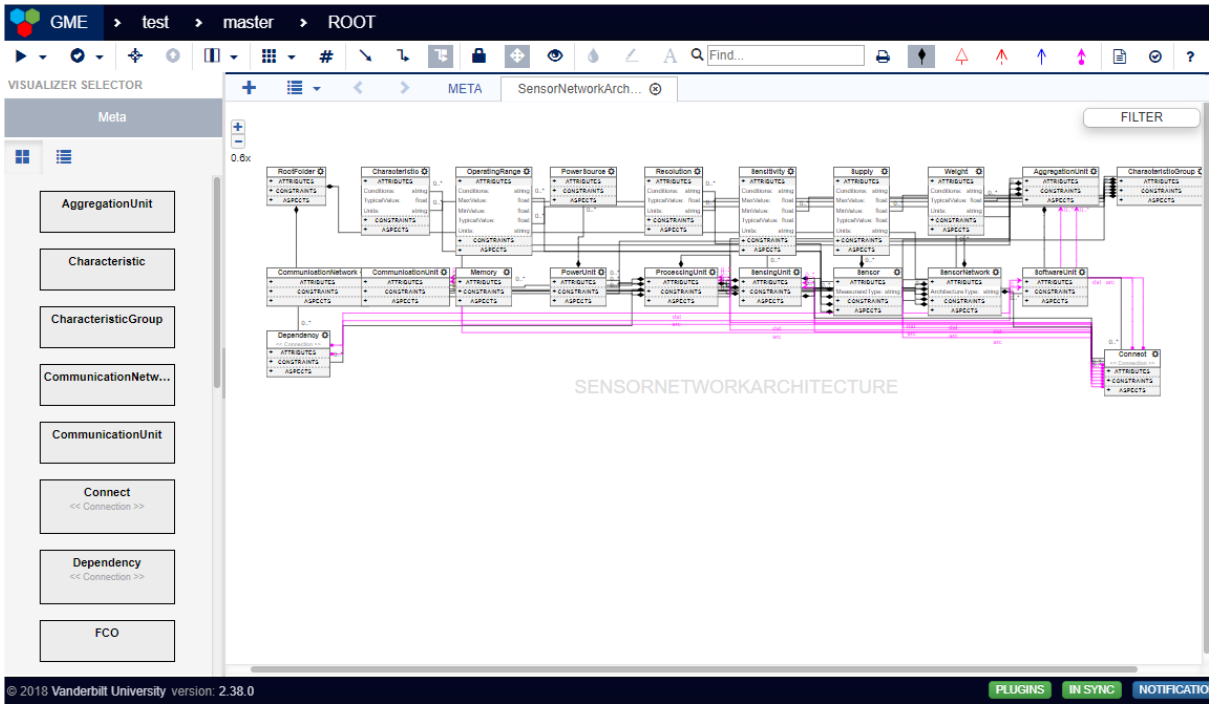

Fig. 5.2 Instrumentul de modelare migrat în Cloud

5.2 Deployment în Cloud - Rezultatele Testelor de Performanță

Pe lângă rezultatele obținute la evaluarea statică, această secțiune prezintă performanța obținută la executarea testelor cu Prometheus & Grafana și Sysdig, pentru implementarea WebGME. Rezultatele pentru instrumentele de monitorizare sunt prezentate în Tabelul 5.1. Aceste valori sunt utilizate pentru măsurarea performanței și indică, de asemenea, utilizarea aplicației. Acestea au fost obținute prin rularea interogărilor Prometheus PromQL. Interogările furnizează date în timp real care ajută utilizatorul să vizualizeze starea resurselor implementate. Aceste rezultate ale testelor sunt obținute după utilizarea normală a aplicației de către doi utilizatori; resursele CPU sunt măsurate în milicore (m). Consumurile CPU pentru podurile Mongo și WebGME sunt prezentate în Tabelul 5.1. O parte din aceste metrici au fost prezentate în [20] și, de asemenea, în [21], este prezentată o soluție de monitorizare pentru dispozitivele cloud-edge.

Table 5.1 Rezultatele monitorizării Prometheus pentru containere

Metrici	Rezultat (interval de o oră)
Container CPU Mongo	2 m ÷ 8 m
Container Memory Mongo	70 MB
Container CPU Webgme	0.5 m ÷ 2 m
Container Memory Webgme	90 MB

Rezultatele monitorizării clusterului sunt prezentate în Tabelul 5.2. Acestea au fost obținute folosind interogările PromQL. Scopul acestor interogări este de a monitoriza starea resurselor clusterului în timpul execuției aplicației. Rezultatele reprezintă consumul mediu pe un interval de timp de o oră.

Table 5.2 Rezultatele metricilor pentru cluster măsurate cu Prometheus

Metrici	Rezultat
Cluster memory usage	47.5 %
Total memory	3.84 GB
Used memory	1.82 GB
Cluster CPU usage	21.34 %
Total CPU	2 cores
Used CPU	0.43 cores

Pentru comparație, Tabelul 5.3 prezintă, de asemenea, rezultatele metricilor prezentate pe în Sysdig. Ele sunt obținute în aceleași condiții ca cele întâlnite în timpul interogării Prometheus PromQL și reprezintă consumurile CPU și memorie pentru containerele Mongo și WebGME; resursele CPU sunt măsurate în cores. Instrumentul de monitorizare Sysdig din IBM Cloud oferă date în timp real privind CPU și memorie pentru containerele mongo și webgme, timp de 1 oră. Consumul mediu de CPU și memorie monitorizat corespunde valorilor din Tabelul 5.3.

Table 5.3 Rezultatele monitorizării Sysdig pentru containere

Metrici	Rezultat (interval de o oră)
Container CPU Mongo	0.01 cores
Container Memory Mongo	63.21 MB
Container CPU WebGME	0.01 cores
Container Memory WebGME	65.75 MB

În ceea ce privește utilizarea memoriei și CPU a clusterului, acestea sunt monitorizate implicit pe consola Sysdig. Deoarece acesta consola nu este personalizata, am obținut procentul de CPU și memorie consumate prezentat în Tabelul 5.4. Rezultatele reprezintă consumul mediu pentru o oră.

Table 5.4 Rezultate Sysdig – metrici cluster

Metrici	Rezultat
Cluster memory usage (%)	30.74 %
Cluster CPU usage (%)	23.63 %

De asemenea, am studiat implementarea altor soluții de monitorizare bazate pe agenți, cum ar fi Dynatrace [22], AppDynamics [23] și NewRelic [24]. Toate aceste soluții oferă capacități multiple și, în anumite privințe, sunt mai capabile decât Prometheus și oferă, de asemenea, o abordare SaaS (Software ca serviciu) pentru consola de monitorizare, dar toate necesită un abonament lunar costisitor deoarece conțin acces la multe caracteristici care nu sunt necesare în configurația curentă. De asemenea, alte platforme [25],[26],[27] care au, de asemenea, un scop educațional și informal ar putea beneficia de astfel de soluții de monitorizare, iar implementarea se poate baza pe ceea ce este prezentat aici.

5.3 Metoda de Evaluare a Algoritmului de Decizie Pentru Opțiunile de Deployment

Algoritmul prezentat este conceput prin determinarea căror dintre rezultatele prezentate este cel mai bun pentru utilizator. Procesul de decizie a fost prezentat și în articolul [28]. În primul rând, estimăm o valoare creată pentru fiecare alegere și o probabilitate pentru fiecare rezultat pe baza numărului de alegeri care duc la un rezultat similar. După cum s-a determinat prin aplicarea algoritmului ID3, caracteristicile cu un câștig mai mare de informații sunt colaborarea și limbajul de programare. Ecuația 2 descrie modul în care se obține valoarea finală, iar Fig. 5.3 și Fig. 5.4 reprezintă arborele de decizie (partea din stânga și din dreapta a arborelui) cu ponderi atașate fiecărui rezultat alături de probabilitatea fiecărei alegeri.

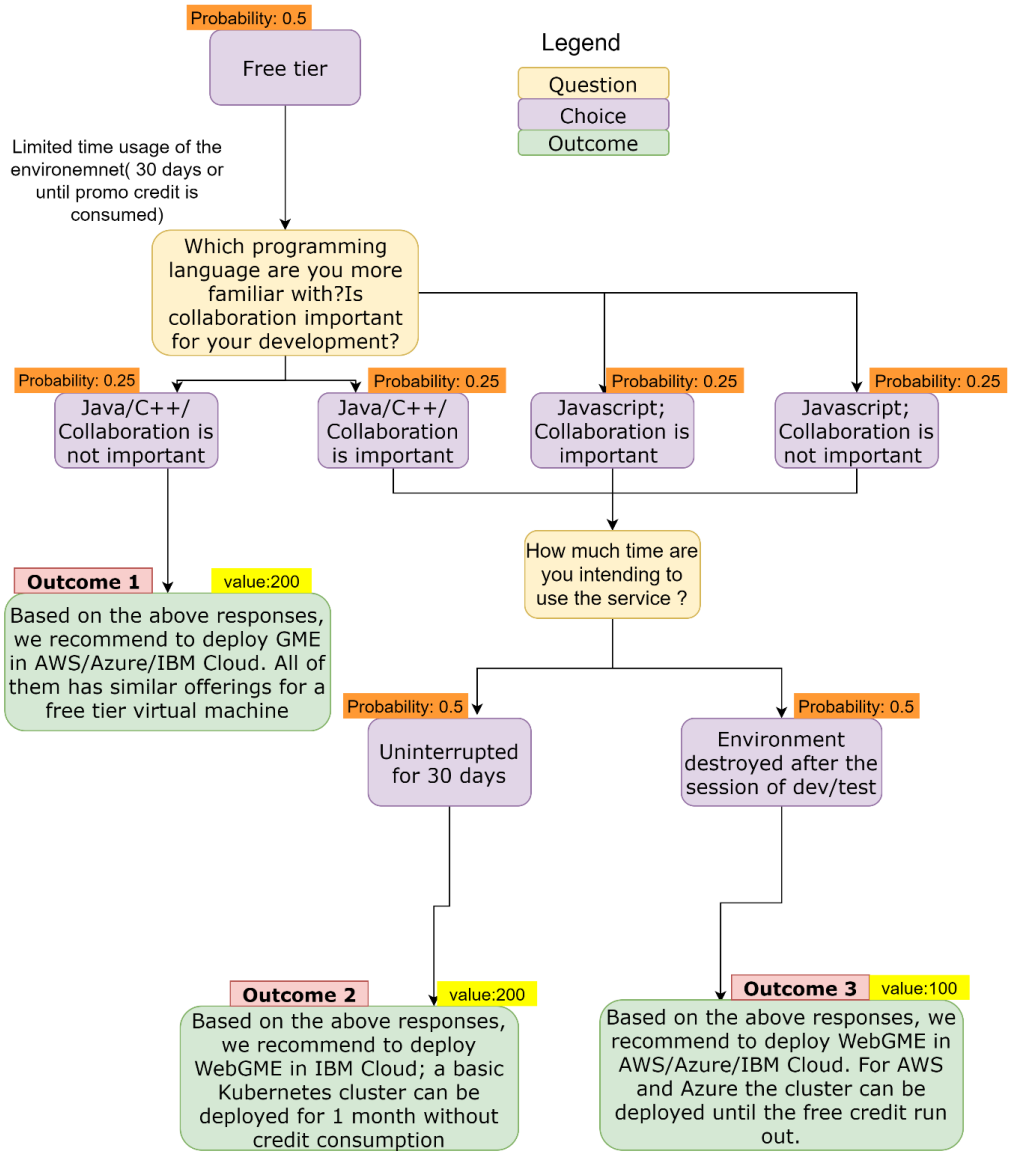

Fig. 5.3 Evaluarea arbore – partea stângă

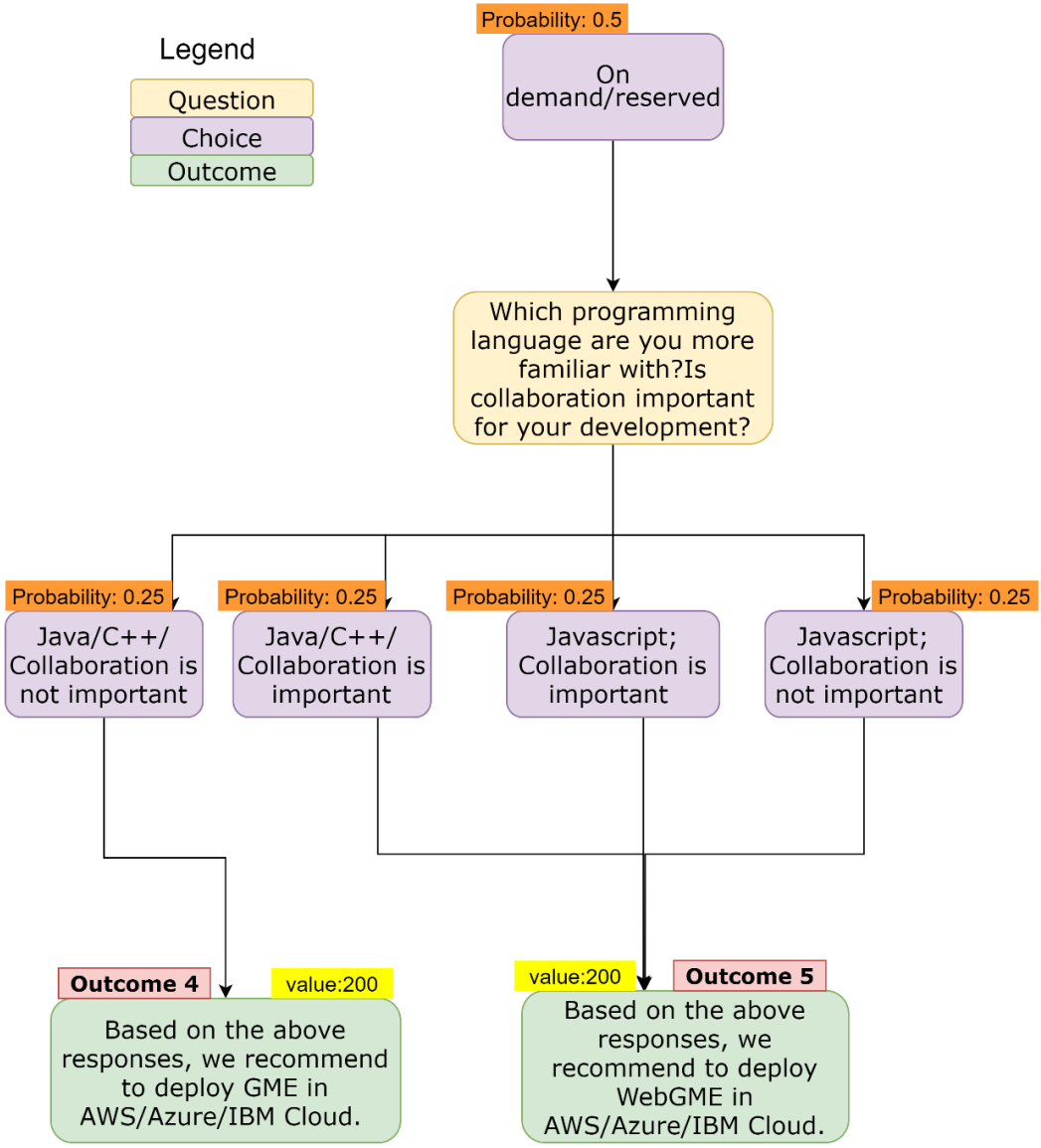

Fig. 5.4 Evaluarea arbore – partea dreaptă

$$Final\ value = (\sum_{n \in N} Probability\ (n)) \times value \qquad (2)$$

unde N reprezintă numărul de opțiuni disponibile care duc la un rezultat pe o cale. De exemplu, pentru Rezultatul 1 (Fig. 5.3), N=2 - două opțiuni conduc la rezultat.

Pentru fiecare rezultat, Tabelul 5.5 arată suma probabilității, valoarea atribuită și valoarea finală, calculate cu ecuația 2. Suma probabilității este suma probabilității tuturor alegerilor care duc la acel rezultat. De exemplu, dacă o întrebare are patru opțiuni, fiecare alegere are o probabilitate de 0,25 să fie aleasă aleatoriu. Pe baza ecuației 2, se evaluează valoarea finală a fiecăruia dintre rezultate. Rezultatele cu cea mai mare valoare sunt cele mai valoroase și cele mai potrivite pentru utilizatori. Valoarea fiecărui rezultat este atribuită în funcție de constrângerile sau lipsa de constrângeri pe care le are fiecare. În acest caz, doar Rezultatului 3 i s-a atribuit o valoare mai mică, deoarece desfășurarea este temporară și există o penalizare

pentru întreruperea serviciului. Ca rezultat, algoritmul evaluează care dintre rezultate este mai valoros pentru utilizator. În plus, alegerile care au o probabilitate combinată mai mare au cel mai mare avantaj, deoarece valoarea atribuită este aceeași pentru majoritatea rezultatelor.

Table 5.5 Evaluarea rezultatelor

Outcome	Value	∑Probability	Final Value
Outcome 1	200	0.75	150
Outcome 2	200	1.75	350
Outcome 3	100	1.75	175
Outcome 4	200	0.75	150
Outcome 5	200	1.25	250

Pe baza acestui calcul, algoritmul determină că cele mai valoroase rezultate sunt cele care sunt rezultatul alegerilor multiple, iar asta înseamnă că se potrivește nevoilor mai multor utilizatori. Acesta este motivul pentru care le consider cele mai potrivite alegeri.

Algoritmul are rolul de a ajuta utilizatorul să decidă opțiunile de deployment. După ce opțiunile de deployment sunt primite de către utilizator, acesta rulează algoritmul, iar implementarea efectivă se va face pe baza detaliilor obținute la pasul de decizie. Partea de decizie are un rol foarte important deoarece este primul pas al implementării: să determine opțiunile pe care le are un utilizator, în funcție de nevoile sale. Desigur, există loc de îmbunătățire din perspectiva îmbunătățirii inteligenței părții de decizie. De exemplu, un utilizator ar putea să-și adauge opțiunile de implementare și un algoritm îmbunătățit, bazat pe învățarea automată [29], ar analiza datele și ar oferi opțiunile utilizatorului.

5.4 Deployment Cloud – Evaluare

Această evaluare se face luând în considerare modelul de calitate prezentat în [30]. Valorile prezentate aici sunt derivate din modelul de calitate a serviciilor și a software-ului. Valorile sunt prezentate în Tabelul 5.6 alături de un scor de la 0 la 3 care reflectă nivelul atributului. Un atribut căruia i se atribuie 0 înseamnă că nu a fost implementat și urmează să fie implementat, 1 ca punctaj înseamnă că există loc de îmbunătățire în acea zonă, 2 înseamnă că este acceptabil, dar poate fi încă îmbunătățit și 3 înseamnă că este aproape perfect.

Table 5.6 Metrici pentru modelul SaaS [30]

Portabilitate	Eficiență	Fiabilitate	Utilizabilitate	Mentenabilitate	Securitate
2	2	1	3	2	1

Potrivit [30], portabilitatea se referă la adaptabilitate și „instalabilitate”. Am atribuit scorul de 2 din cauza ușurinței instalării utilizând platforma de implementare și a modului în care poate fi adaptată și implementată în mai multe medii cloud. Desigur, aici este loc de îmbunătățire.

Eficiența se referă la utilizarea resurselor și, în acest caz, am demonstrat anterior că, consumul de resurse rămâne scăzut în timpul utilizării software-ului. Acest lucru este la fel și atunci când interpretorii sunt implementati. Acest lucru poate fi reglat prin impunerea limitelor de memorie și CPU care pot controla consumul de resurse și îl pot limita la intervalul de consum prevăzut.

Conform [30] fiabilitatea se referă la maturitatea software-ului și recuperabilitatea. Modul de deployment nu este suficient de matur, dar software-ul în sine care este implementat este deja testat. În ceea ce privește interpretorii, aceștia nu sunt suficient de maturi până când nu sunt testați și dezvoltați corespunzător pentru o perioadă mai lungă. În ceea ce privește recuperabilitatea, nu există o metodă de backup pentru datele implementate. Deci, mediul poate fi redistribuit în caz de eșec, dar datele se pierd și nu sunt recuperate. Deci, aceste aspecte trebuie îmbunătățite.

În ceea ce privește uzabilitatea, platforma este ușor de utilizat, dar și mediul de modelare. O secțiune de tutorial poate fi o îmbunătățire, dar acum este înlocuită cu secțiuni de ajutor care sunt prezente pe platformă.

Conform [30] mentenabilitatea se referă la stabilitate, schimbare (se referă la modificarea codului) și testabilitate. Platforma poate fi întreținută foarte ușor și în ceea ce privește modificarea codului, front-end-ul este modular si bazat pe componente și servicii, astfel încât o altă caracteristică poate fi implementată cu ușurință, sau o caracteristică mai veche poate fi abandonată. Aceeași discuție este pentru backend, deoarece un alt controler poate fi implementat sau abandonat. În ceea ce privește testabilitatea, instrumentele de testare pot fi implementate pe clusterul Kubernetes pentru a testa mediul. Desigur, acest lucru poate fi îmbunătățit cu o secțiune de administrare cu mai multe funcții care pot ajuta utilizatorul să finalizeze sarcinile mult mai ușor.

În ceea ce privește securitatea, este nevoie de o soluție care să scaneze vulnerabilitățile pe clusterul Kubernetes și să scaneze imaginile Docker. Imaginile docker sunt deja scanate atunci când sunt introduse în IBM CLoud Container Registry și pot fi scanate și pe Dockerhub dacă se face upgrade la un abonament pro. De asemenea, politica de securitate a podului din Kubernetes poate fi încă configurată (învechită în Kubernetes versiunea 1.21; va fi eliminată în versiunea 1.25 și înlocuită cu o altă soluție) pentru a restricționa crearea și actualizarea podului. În mod similar, în articolul [31], este prezentată o comparație a soluțiilor de monitorizare a cloud-edge și evidențiază avantajele unei astfel de implementări și dispozitive cloud.

5.5 Discuţie

5.5.1 Instrumente de modelare

Infrastructura cloud de microservicii pentru WebGME a fost implementată folosind Kubernetes și Docker. Ca rezultat, WebGME poate fi accesat privat printr-un IP public al Kubernetes și gestionat privat pe un Cloud public. Pentru aceste activități au fost realizate două deploymenturi și două servicii noi.

A fost introdusă o nouă configurație pentru imaginea docker WebGME, pentru a se potrivi conectivității MongoDB. Se așteaptă ca WebGME în Cloud să funcționeze mai bine decât alte soluții desktop și bazate pe web (WebGME clasic și AToMPM), conform următoarelor criterii: disponibilitate ridicată, recuperare în caz de dezastru, colaborare privată, management al resurselor și mediu privat de modelare. Cu toate acestea, acestea sunt influențate de ofertele

IBM Cloud, AWS și Azure și de performanța lucrătorilor. Disponibilitate ridicată înseamnă capacitatea unui sistem/aplicație de a rula fără defecțiuni pentru o perioadă mare de timp. Pentru munca noastră, recuperarea în caz de dezastru a fost considerată capacitatea sistemului de a fi încă disponibil dacă are loc un dezastru natural/non-natural. Colaborarea privată și mediul privat de modelare se referă la faptul că, Git este accesibil și vizibil pentru anumiți utilizatori. Managementul resurselor este un criteriu care este moștenit din implementarea Cloud, unde resursele pot fi implementate/șterse după cum este necesar.

Toate criteriile prezentate reprezintă punctele forte pe care le are implementarea privată a WebGME în Cloud în comparație cu instrumente similare precum WebGME clasic și AToMPM. Aceste îmbunătățiri pot ajuta echipele să atingă obiectivele de lucru în echipă și managementul mediului privat, alături de economiile de costuri garantate de planurile de costuri ale furnizorului de cloud.

5.5.2 Analiza complexității

Rezultatele din Secțiunea 4.3.1 arată că, complexitatea ciclomatică pentru deploymentul automat este 1, în timp ce pentru deploymentul manual este 3. După cum era de așteptat, metoda automată are o valoare mai mică a complexității ciclomatice decât cea manuală. Implicația acestei constatări este importantă pentru scopul meu, deoarece este un prim pas pentru a demonstra că dezvoltarea unei platforme de deployment automat ar face o diferență semnificativă. O valoare mai mică a complexității indică mai puține erori, un proces mai ușor de înțeles de către utilizator, mai ușor de testat și mai ușor de întreținut.

Pentru cele două procese de deployment (reprezentate în Fig. 4.1 și Fig. 4.2), rezultatele complexității Yaqin (YC) prezentate în Secțiunea 4.3.1 diferă de asemenea. În primul rând, observăm că primul proces, pentru implementarea manuală, este mai complex decât implementarea automată din cauza diferenței de număr de ramuri. Cu toate acestea, acesta nu este singurul motiv; este unul dintre punctele forte ale metodei propuse în [14], deoarece metrica Yaqin implică mulți parametri și este foarte sensibilă în găsirea diferențelor dintre numerele și tipurile ramurilor (OR, AND și XOR). Rezultatele prezentate ne-au determinat să observăm că, complexitatea generală Yaqin este mai mult decât dublă pentru procesul de deployment manual decât pentru cel automat. Acest rezultat confirmă avantajul unei platforme de implementare și simplitatea ei de utilizare în comparație cu metoda manuală.

Ca o evaluare suplimentară, să luăm în considerare și complexitatea sarcinilor umane (CH) estimate în Secțiunea 4.3.1. Aceasta este mai specifică domeniului tezei noastre, iar atribuirea ponderilor este motivată pe baza cunoștințelor tehnice necesare pentru fiecare sarcină. Ponderile atribuite sunt între 1 și 3, cu valoarea 3 pentru sarcinile cele mai complexe. În cazul deploymentului automat, nici unei singure sarcini nu i sa acordat o pondere de 3, spre deosebire de deploymentul manual. Conform acestei metrici, CH este 20 pentru deploymentul manual și doar 8 pentru cel automat, adică de aproape 2 până la 3 ori mai mic. Aceasta este o diferență semnificativă, deși o platformă de deployment permite o reducere importantă a muncii manuale, lărgind și gama de oameni care pot implementa mediul software de care au nevoie în cloud, fără a fi nevoie de cunoștințe tehnice foarte specializate în cloud computing. Rețineți că metrica CH definită în secțiunea 4.3.1. este similar cu metrica YC, deoarece folosește și ponderea cognitivă pentru a evalua dificultatea de înțelegere a software-ului. Cu metrica specifică, personalizez această metodă la sarcinile necesare pentru pregătirea/implementarea/utilizarea unui mediu. Criteriile alese pot fi aplicate și altor procese

similare, care necesită diverse tipuri de interacțiune cu utilizatorul pentru a efectua sarcini manuale. Această măsurătoare poate fi utilizată și pentru alte medii software [32] implementate în cloud.

Multicloud deployment complexity

Pentru implementarea pe alte platforme cloud, pașii prezentați în Fig. 4.1 și Fig. 4.2 sunt similari, cu câteva diferențe în ceea ce privește ofertele specifice furnizorului de cloud. De exemplu, pentru deploymentul manual, sarcina 3 din Fig. 4.1 ar trebui să implementeze metode specifice de conectare Azure sau AWS, cum ar fi ID-ul principal al serviciului și parola, sau ID-ul cheii de acces AWS și cheia de acces secretă. Sarcina 5 ar fi înlocuită cu instalarea CLI-ului pentru cloudul respectiv, iar sarcina 6 s-ar conecta la platforma cloud aleasă. Pentru sarcina 8, există mai multe alternative în funcție de preferințele utilizatorului. De exemplu, se poate utiliza AWS Elastic Container Registry, Azure Container Registry sau Docker Hub. Desigur, sarcina 12 s-ar modifica și pentru a necesita conexiunea la platforma cloud vizată. Pentru procesul de deployment automat, sarcina 2 din Fig. 4.2 ar cere utilizatorului să preia datele de autentificare specifice AWS și Azure; sarcina 5 devine selecția cloudului vizat, iar sarcina 6 ar crea conexiunea la cloudul selectat de pe platforma multicloud. Prin urmare, pașii sunt ușor diferiți pentru a se potrivi furnizorului de cloud și nu sunt modificați în totalitate. Valoarea complexității rămâne aceeași pentru deploymentul în alte platforme cloud.

5.5.3. Analiza performanței

Pentru testarea performanței, am folosit Prometheus și Sysdig pentru monitorizarea clusterului Kubernetes și a containerelor implementate. Valorile CPU și memoriei containerului indică utilizarea unei aplicații și eficiența acesteia. Dacă o aplicație este ineficientă și memoria crește la un nivel alarmant pentru cluster, poate apărea un semnal OOM (Out of Memory) pe logurile respectivilor workeri; într-un astfel de caz, podul care consumă prea multă memorie este evacuat și realocat. Dacă graficul arată un consum alarmant de memorie și CPU, limitele de memorie trebuie să devină mai mari decât cererea de memorie, dar nu la fel de mari ca întreaga memorie a worker-ului.

După calcularea acestor metrici pentru mediul nostru de testare (a se vedea Tabelul 5.1) și pe baza memoriei clusterului și a utilizării CPU din Tabelul 5.2 și Tabelul 5.4, consider că dimensionarea actuală este capabilă să susțină mediul de modelare WebGME alături de soluția de monitorizare. Consumul de memorie al pod-urilor Kubernetes poate fi controlat prin introducerea de memory si CPU requests și a limitelor in definiția containerului (deployment). Acei parametri de configurare au rolul de a împiedica containerul să folosească mai multe resurse decât este destinat. Când limita este atinsă, containerul este repornit [33]. Cu toate acestea, consumul de resurse poate crește dacă mulți utilizatori utilizează în mod activ același serviciu, adică accesează mediul de modelare ca serviciu în același timp. În comparație cu aceste constatări, obținute cu Prometheus & Grafana, rezultatele Sysdig din Tabelul 5.4 sunt relativ similare. Ambele teste returnează rezultate sub 100 MB pentru memoria consumată, atât pentru WebGME, cât și pentru Mongo. Din cauza limitărilor resurselor clusterului Kubernetes utilizate în mediul nostru de testare, acest lucru nu poate fi testat în același timp. Diferența în cazul CPU este că unitatea de măsură folosită în ambele teste este diferită. În consola de monitorizare Sysdig, nucleele sunt folosite ca unități de monitorizare a CPU, în timp ce pe Grafana a fost aleasă miimea de nucleu (m). Prin urmare, abordarea utilizată în Grafana este

mai sensibilă la consumul mai scăzut al procesorului în comparație cu aproximarea de bază utilizată de Sysdig.

Rezultatele din Tabelul 5.2 și Tabelul 5.4 demonstrează un consum mai mic de memorie atunci când Prometheus și Grafana nu sunt instalate pe cluster, deși consumul CPU este similar. Acest lucru poate fi considerat un avantaj al monitorizării Sysdig IBM Cloud, alături de faptul că nu necesită configurare din partea utilizatorului. Dezavantajul este că Sysdig nu este prezent sub această formă în alte platforme cloud, precum Azure și AWS.

5.5.4 Platforma de deployment automat

Cercetarea prezentată a arătat că un deployment automat ar face o diferență importantă în accesibilitatea unui mediu software furnizat ca serviciu în cloud. Platforma rezultată pentru implementarea automată a cloud-ului este scrisă în Node.js pentru partea de backend și în Angular pentru frontend. Platforma este concepută pentru a implementa mai multe tipuri de medii software în cloud cu furnizori diferiți, cu posibilitatea de a adăuga altele noi pe listă.

Platforma este capabilă să automatizeze implementarea mediului de modelare și metamodelare WebGME în IBM Cloud, AWS,Azure si OpenShift Online. În prezent, implementarea este realizată folosind tehnologii precum Docker, Kubernetes și Helm; au fost aleși pentru că funcționează bine și cu alți furnizori importanți de cloud, nu numai cu IBM Cloud. Pentru a începe implementarea, un utilizator trebuie să se înregistreze pe platformă, să creeze un proiect, să îl seteze ca activ și să înregistreze detaliile de conectare la cloud în forma necesară pentru furnizorul de ales. Ca lucrări viitoare, platforma va sprijini și implementarea serviciului Kubernetes; în funcție de furnizorul de cloud ales, utilizatorul va avea un tutorial despre cum să implementeze un serviciu Kubernetes pe platforma cloud.

Astfel, platforma de deployment automat a mediului software în cloud oferă utilizatorilor non-tehnici posibilitatea unei setări mai ușoare și a unei implementări rapide a mediului software necesar în cloud. De asemenea, este posibil să se creeze mai multe medii pentru testare, dezvoltare și producție ori de câte ori este nevoie. Mediul software implementat poate fi, de asemenea, șters automat de către utilizator prin intermediul platformei. Acest tip de platformă s-a dovedit a fi foarte util în vremuri de pandemie, așa cum este descris în articolele [34] și [35].

6. Concluzie

În primul rând, teza a prezentat stadiul actual pe subiecte precum modelarea ca serviciu, deploymentul în cloud și analiza performanței, care se potrivesc firesc pentru acest tip de cercetare. Scopul a fost realizarea unei platforme de deployment multicloud care să-și poată ajuta utilizatorii, reprezentați de studenți, profesori și cercetători, să implementeze un mediu software în multicloud, chiar și fără cunoștințe tehnice vaste. Ca urmare, obiectivele specifice au fost următoarele: conceperea și realizarea unei noi platforme multicloud, evaluarea metodelor de deployment și compararea între deploymentul automat și cel manual pentru a înțelege nevoile unei astfel de platforme din punct de vedere al complexității. De asemenea, platforma s-a dezvoltat în direcția de a ajuta utilizatorul să decidă detaliile de deployment și de a implementa interpretoare de modele alături de mediul de modelare. Ca urmare, teza a prezentat metodele pentru fiecare dintre obiectivele propuse și platforma de implementare multicloud rezultată. În ceea ce privește complexitatea, scopul meu a fost să prezint un set de metrici de complexitate pentru procesele de business care reprezintă procedura de deployment manual a mediului de modelare pe un cluster Kubernetes și deploymentul automat de pe platforma multicloud. Acest calcul a fost realizat pentru a identifica un motiv adecvat pentru dezvoltarea platformei multicloud. Având în vedere această perspectivă, am studiat articole care m-au ajutat să aleg metricile potrivite pentru cazul meu. Articolele au fost foarte explicite în ceea ce privește formulele utilizate pentru metrici și motivul pentru care sunt utilizate. Ca urmare, am ales să folosesc două metrici de complexitate: complexitatea ciclomatică, care este foarte robustă, și metrica Yaqin, care este mai complexă în comparație cu prima, deoarece conține multe ecuații care au rolul de a crește precizia rezultatului. Folosind aceste metrici, am ajuns la concluzia că deploymentul automat este de aproape trei ori mai puțin complexă decât deploymentul manual și necesitatea unei platforme multicloud este într-adevăr binevenită. În cele din urmă, am propus o nouă metrică care se bazează pe dificultatea sarcinilor. Criteriile de atribuire a ponderilor se bazează pe cantitatea de cunoștințe necesare unui utilizator/dezvoltator pentru a finaliza o sarcină. Pentru fiecare dintre sarcini, am descris motivul din spatele ponderii cu un exemplu real despre cum se o realizează. Ponderea dată a fost între 1 și 3, cele ce au valoarea 3 fiind sarcinile cele mai complexe. În cazul implementării automate cu platforma, nici unei singure sarcini nu i s-a acordat o pondere de 3, spre deosebire de deploymentul manual în care scorul de complexitate a fost mai mult decât dublu.

Un alt aspect investigat este performanța serviciului. Am comparat și implementat două soluții de monitorizare pentru mediu: Prometheus cu Grafana și IBM Cloud monitoring (Sysdig). După configurarea mediului, am început prin a folosi limbajul PromQL care interogează baza de date Prometheus pentru metrici. Am lucrat mai întâi direct cu consola web Prometheus și apoi mi-am creat vizualizări în Grafana. Am descris în detaliu interogările utilizate pentru monitorizarea următorilor parametri: utilizarea CPU și a memoriei pentru containerele Mongo și WebGME, namespace, podul și starea deploymentului și, în final, consumul de resurse la nivel de cluster Kubernetes. În continuare, am implementat soluția de monitorizare IBM Cloud Monitoring care se bazează pe Sysdig. Am comparat ambele soluții și, deși soluția de monitorizare IBM Cloud este mai ușor de configurat, are dezavantajul că este implementabilă doar pe IBM Cloud (pentru o soluție multicloud, nu mă pot baza pe servicii diferite) și, de asemenea, are un abonament lunar scump care în acest caz nu poate justifica costul, deoarece Prometheus și Grafana pot oferi rezultate similare gratuite.

Teza a propus și un nou algoritm de decizie pentru alegerea opțiunilor de deployment pe care le oferă platforma. Acest algoritm are rolul de a ajuta utilizatorul să decidă care este cel mai potrivit mediu de modelare și cloud pentru deployment. În primul rând, am prezentat un set de date cu opțiuni de deployment și am aplicat algoritmul ID3 setului de date pentru a crea un arbore de decizie. Algoritmul ID3 a fost aplicat pentru a prezice condițiile de implementare a GME și WebGME. Pentru această parte, furnizorul Cloud a fost omis. După acest experiment, am creat un algoritm de decizie care ia în considerare toate detaliile de implementare. Ca rezultat al algoritmului ID3, căile au fost, de asemenea, ghidate de aceiași factori de decizie: colaborare și limbaj de programare alături de noi completări, cum ar fi perioada de disponibilitate a clusterului. După elaborarea algoritmului, acesta a fost integrat și în platforma multicloud.

Teza a descris și scenariile privind modul în care această soluție poate fi clasificată. S-a prezentat de ce soluția poate fi considerată în același timp PaaS sau SaaS în funcție de punctul de vedere al utilizatorului. De asemenea, au fost prezentate opțiunile care au fost luate în considerare pentru dezvoltarea și implementarea interpretorului de modele. A doua variantă care presupune încapsularea interpretorului în imaginea Docker a fost aleasă datorită avantajelor pe care le are. De asemenea, a fost descrisă noua adăugare la Dockerfile pentru ca această implementare să funcționeze. De asemenea, au fost descrise câteva metode de evaluare a obiectivelor pentru a demonstra utilitatea acestora sau dacă metoda este eligibilă pentru a fi utilizată în acest scenariu.

6.1 Rezumatul contribuțiilor originale

O listă a contribuțiilor originale ale acestei teze este prezentată mai jos.

1. Un prim obiectiv al acestui studiu a fost identificarea referințelor necesare care vor ajuta la obținerea cunoștințelor necesare în domeniu și la implementarea proiectului final. Am selectat articole care descriu subiecte precum modelare și cloud, algoritmi decizionali, metrici cloud și metrici de complexitate. Acestea au fost alese pe baza descrierii elementelor teoretice și a exemplelor practice. Principala contribuție în acest sens o reprezintă lecțiile învățate din temele prezentate.
2. Implementarea mediului de modelare WebGME în IBM Cloud, însoțită de un exemplu de migrare a metamodelului dintr-un metamodel sursă pentru rețele de senzori, dezvoltat inițial în GME - varianta on-site a acestui mediu. WebGME în cloud este accesibil tuturor membrilor unei echipe de dezvoltare în același timp, iar modificările pe același proiect pot fi făcute de aceștia pe orice dispozitiv.
3. Comparație între WebGME implementat în cloud cu soluțiile tradiționale și alte instrumente de modelare. Prezentarea avantajelor unei astfel de implementări și a necesității acesteia.
4. Am propus o arhitectură cloud detaliată pentru platforma de implementare multicloud.
5. Elaborarea unui user story detaliat care să prezinte situația în care este nevoie de o astfel de platformă.
6. Prezentarea ideii din spatele implementării și vizualizarea cazului de utilizare pentru platformă, din perspectiva utilizatorului.
7. Am propus un algoritm de decizie care poate ajuta utilizatorul să decidă ce mediu de modelare să implementeze (WebGME sau GME) și în ce cloud.

8. Am reprezentat modele detaliate de proces pentru deploymentul automat în cloud și deploymentul manual în cloud.
9. Alegerea tehnologiilor utilizate pentru infrastructură. Alegerea tehnologiilor reprezintă una dintre contribuțiile arhitectului de Infrastructură/Cloud/Aplicație într-un proiect comercial.
10. Proiectarea aplicației multicloud din perspectiva font-end.
11. Caracteristici elaborate pentru aplicația care are rolul de a se potrivi nevoilor utilizatorului.
12. Prezentarea metodelor de implementare pentru backend și frontend.
13. Identificați schimbările în mediul educațional cauzate de pandemia COVID-19.
14. Propunerea unei soluții tehnice la probleme precum dificultățile de colaborare, nevoia de economisire a costurilor și un management mai eficient al resurselor. Teza a prezentat o platformă care poate ajuta la depășirea limitărilor prezentate și are rolul de a implementa instrumente de modelare, cum ar fi instrumentul de modelare WebGME, în cloud.
15. Dezvoltarea unei platforme care promovează ideea de multicloud ca înlocuitor pentru mediul de laborator în domeniul modelării. Ușurința de utilizare a platformei a fost, de asemenea, un accent și o contribuție; platforma poate fi utilizată fără cunoștințe tehnice ca o condiție prealabilă.
16. Am oferit o comparație solidă între o metodă tradițională de deployment care necesită cunoștințe tehnice vaste și utilizarea platformei nou dezvoltate.
17. Am aplicat metricile de complexitate studiate asupra procesului de deployment am comparat rezultatele.
18. Am propus o nouă metrică de complexitate centrată pe dificultatea sarcinilor. Criteriile de atribuire a ponderii se bazează pe cantitatea de cunoștințe necesare unui utilizator/dezvoltator pentru a finaliza o sarcină și pe o descriere completă a tuturor motivelor din spatele atribuirii ponderii.
19. Am furnizat două soluții de monitorizare pentru mediul implementat.
20. Compararea instrumentelor de monitorizare pe baza rezultatelor performanței, caracteristicilor și costurilor.
21. Analiza detaliată a rezultatelor pentru metricile complexității procesului de afaceri.
22. Analiza detaliată a performanței instrumentelor de monitorizare și a metricilor clusterului.
23. Am oferit o discuție detaliată a rezultatelor în contextul implementării manuale și automate și a performanței acestora.
24. Prezentarea și implementarea unei caracteristici pentru platforma de implementare sub forma unui algoritm decizional care are scopul de a ajuta utilizatorul să aleagă detaliile pentru implementarea cloud. Crearea unui set de date bazat pe nevoile identificate ale utilizatorilor. Am aplicat algoritmul ID3 pe setul de date care conține o combinație de decizii de implementare pentru a prezice căile. Având în vedere arborii rezultați (după aplicarea algoritmului ID3), mi-am creat algoritmul de decizie unde am adăugat și celelalte opțiuni de implementare. Algoritmul de decizie conduce la 5 rezultate.
25. Am evaluat algoritmul determinând care dintre rezultate este cel mai bun pentru utilizator. Această evaluare a fost făcută pe baza probabilității de alegere și a unei valori atribuite rezultatului (pe baza constrângerilor pe care le are fiecare rezultat).
26. Implementarea algoritmului de decizie în platforma multicloud.

27. Am analizat soluția din punctele de vedere PaaS și SaaS și am concluzionat că pot fi ambele în același timp.
28. Am dezvoltat două soluții și a prezentat arhitectura pentru ambele privind strategia de implementare a interpretorului de modele alături de instrumentul de modelare.
29. Implementarea soluției în platforma multicloud (AWS, Azure, IBM Cloud și OpenShift Online).
30. Am evaluat metoda de deployment in cloud pe baza unui model de calitate pentru SaaS.

6.2 Listă Publicații

Pentru această teză, am folosit conținutul articolelor 1,4,6 și 9 din lista de mai jos.

1. **Marian Lacatusu**, A. D. Ionita, Florin Lacatusu and I. Damian, "Decision support for multicloud deployment of a modeling environment," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 247-251, doi: 10.1109/ICCP56966.2022.10053951

2. Florin Lacatusu, A. D. Ionita, **Marian Lacatusu**, I. Damian and D. Saru, "A Comparison of Cloud Edge Monitoring Solutions for a University Building," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 253-257, doi: 10.1109/ICCP56966.2022.10053978

3. Ioan Damian, **Marian Lacatusu**, Florin Lacatusu, Anca Daniela Ionita, Web Services for Guiding Persons with Locomotor Impairments in Public Spaces, *2022 26th International Conference on System Theory, Control and Computing* (ICSTCC), 2022, pp.639-644, IEEE, INSPEC, doi:10.1109/ICSTCC55426.2022.9931861, **WOS:000889980600107**

4. **Lăcătușu, Marian**; Ionita, A.D.; Anton, F.D.; Lăcătușu, Florin Analysis of Complexity and Performance for Automated Deployment of a Software Environment into the Cloud. *Appl. Sci.* **2022**, *12*, 4183. https://doi.org/10.3390/app12094183, *Impact factor:* 2.84 - **Q2**, **WOS:000794735600001**

5. Lăcătușu, Florin; Ionita, A.D.; **Lăcătușu, Marian**; Olteanu, A. Performance Evaluation of Information Gathering from Edge Devices in a Complex of Smart Buildings. *Sensors* **2022**, *22*, 1002. https://doi.org/10.3390/s22031002, *Impac factor:* 3.84 – **Q2**, **WOS:000760176500001**

6. **Marian Lacatusu**, Florin Lacatusu, Ioan Damian, Anca Daniela Ionita, Multicloud Deployment to Support Remote Learning, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4601-4606, IATED Digital Library, doi: 10.21125/inted.2021.0936

7. Ioan Damian, **Marian Lacatusu**, Anca Daniela Ionita, Florin Lacatusu, Software Services to Support Faculty Management in Times of Pandemic, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4634-4639, IATED Digital Library, doi: 10.21125/inted.2021.0943

8. Florin Lacatusu, I. Damian, A. D. Ionita and **Marian Lacatusu**, Smart Building Manager Software in Cloud, *University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 83, no. 4,* 2021., *Impact factor:* 0.37, **WOS:000741473700003**

9. **Marian Lacatusu** and A. D. Ionita, "Metamodeling environment in Cloud," *University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 82, no. 3,* 2020, *Journal:* U.P.B. Sci. Bull Series C, *Impact factor:* 0.37, **WOS:000557847800003**

10. Olteanu, Adriana**; Lacatusu, Marian**; Ionita, Anca Daniela; Lacatusu, Florin, ”Platform for Informal Education and Social Networking to Increase Awareness Regarding Nuclear Vulnerabilities” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 333-340. **WOS:000467466800045**

11. Adriana Olteanu, Florin Lacatusu, **Marian Lacatusu**, Iulian Craciun, Anca Daniela Ionita, ”Mobile Application for Crisis Situations in a University Campus” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 280-287. **WOS:000467466800038**

6.3 Perspective de viitor

Dintr-o perspectivă viitoare, adoptarea platformei multicloud pentru a fi utilizată într-un mediu universitar este continuarea cea mai firească. Profesorii, studenții și experții în modelare pot folosi platforma așa cum este (din perspectiva caracteristicilor).

În ceea ce privește îmbunătățirile tehnice, implementarea serviciului Kubernetes poate fi adăugată ca o caracteristică alături de implementarea mediilor de modelare și a interpretorilor acestora. Pentru a obține implementarea infrastructurii direct de pe platformă, componenta de preț trebuie explicată utilizatorilor, precum și opțiunile pe care le are pentru implementare. Acest lucru necesită un cluster Kubernetes cu un master și un worker, iar timpii de implementare ar trebui să difere între cei trei furnizori de cloud acceptați (IBM Cloud, AWS, Azure) și OpenShift online. Cu o astfel de caracteristică, utilizatorul are nevoie doar de un abonament activ la cloud ca o condiție prealabilă. Aceasta poate fi o completare bună și o modalitate de a simplifica sarcinile manuale rămase pe care trebuie să le facă un utilizator.

În viitor, alte opțiuni de deployment pot fi adăugate la catalogul actual pentru ca platforma să fie mai versatilă. De exemplu, alte medii software pot fi adăugate la catalogul de deployment. Astfel, platforma de implementare automată a mediului software în cloud oferă utilizatorilor non-tehnici posibilitatea unei setari mai ușoare și a unei implementări rapide a mediului software necesar în cloud. De asemenea, este posibil să se creeze mai multe medii pentru testare, dezvoltare și producție ori de câte ori este nevoie. Mediul software implementat poate fi, de asemenea, șters automat de către utilizator prin intermediul platformei. Deci, furnizorii de cloud acceptați pot fi, de asemenea, îmbunătățiți prin adăugarea de suport pentru noi furnizori, cum ar fi Google Cloud, Alibaba Cloud și multe alte exemple.

În ceea ce privește partea de decizie, un utilizator ar putea să-și adauge opțiunile de implementare și un algoritm îmbunătățit, bazat pe învățarea automată [29], ar analiza datele și ar oferi opțiunile utilizatorului.

Bibliografie selectivă

[1]. Tomarchio, O.; Calcaterra, D.; Modica, G.D. Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks, *J Cloud Comp,* **2020,** vol 9, issue 49. https://doi.org/10.1186/s13677-020-00194-7

[2]. Giannakopoulos, I.; Konstantinou, I.; Tsoumakos, D. . Cloud application deployment with transient failure recovery. *J Cloud Comp*, **2018,** vol. 7, issue 11. https://doi.org/10.1186/s13677-018-0112-9

[3]. S. Jung and J.-H. Huh, "An Efficient LMS Platform and Its Test Bed," *Electronics*, pp. 8, 154., 2019.

[4]. W. Ali, "Online and Remote Learning in Higher Education Institutes: A Necessity in light of COVID-19 Pandemic.," *Higher Education Studies*, pp. 10 ,16, 2020.

[5]. GME, Generic Modeling Environemnt, Accessed 25 November, 2020., Retrieved from: http://www.isis.vanderbilt.edu/projects/GME

[6]. A. D. Ionita, F. Anton and A. Olteanu, "A. Sensor Network Modeling as a Service," *In Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018),* pp. 346-353, 2018.

[7]. A. D. Ionita, "Research Experience of Master's Students for Modeling Hazard Management Systems," *Proceedings of INTED2017 Conference 6th-8th March 2017, Valencia, Spain,* pp. 8865-8870, 2017.

[8]. S. Popoola, J. Carver and J. Gray, "Modeling as a Service: A Survey of Existing Tools," *in Proceedings of the Model-Driven Engineering Languages and Systems Conference*, 2017

[9]. WebGME, WebGME Web page, Accessed 30 November, 2020., Retrieved from: https://webgme.org/

[10]. Li, G.; Woo, J.; Lim, S.B. HPC Cloud Architecture to Reduce HPC Workflow Complexity in Containerized Environments. *Appl. Sci.* **2021**, *11*, 923. https://doi.org/10.3390/app11030923

[11]. Shah, S.A.R.; Waqas, A.; Kim, M.-H.; Kim, T.-H.; Yoon, H.; Noh, S.-Y. Benchmarking and Performance Evaluations on Various Configurations of Virtual Machine and Containers for Cloud-Based Scientific Workloads. *Appl. Sci.* **2021**, *11*, 993. https://doi.org/10.3390/app11030993

[12]. Bystrov, O.; Pacevič, R.; Kačeniauskas, A. Performance of Communication- and Computation-Intensive SaaS on the OpenStack Cloud. *Appl. Sci.* **2021**, *11*, 7379. https://doi.org/10.3390/app11167379

[13]. Shepperd, M. A Critique of Cyclomatic Complexity as a Software Metric, Software Engineering Journal, 1988, pp. 30-36

[14]. Yaqin, M. A.; Sarno, R.; Rochimah, S. Measuring Scalable Business Process Model Complexity Based on Basic Control, *International Journal of Intelligent Engineering and Systems 13* (2020): 52-65

[15]. Prometheus Overview. Available online: https://prometheus.io/docs/introduction/overview/ (accessed on 5 April 2021).

[16]. Prometheus configuration Kubernetes. Available online: https://devopscube.com/setup-prometheus-monitoring-on-kubernetes/ (accessed on 1 May 2021).

[17]. M. Lacatusu and A. D. Ionita, "Metamodeling environment in Cloud," University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 82, no. 3, 2020, Journal: U.P.B. Sci. Bull Series C

[18]. M. Kenyeres, and J Kenyeres: "Multi-Sensor data fusion by average consensus algorithm with fully-distributed stopping criterion: comparative study of weight designs.", UPB Scientific Bulletin, Series C: Electrical Engineering and Computer Science, vol. 81, iss. 2, 2019, pp. 27-42

[19]. MetaGMEParadigmImporter: https://github.com/webgme/webgme-engine/tree/master/src/plugin/coreplugins/MetaGMEParadigmImporter, Accessed March 26, 2020

[20]. Lăcătușu, M.; Ionita, A.D.; Anton, F.D.; Lăcătușu, F. Analysis of Complexity and Performance for Automated Deployment of a Software Environment into the Cloud. *Appl. Sci.* **2022**, *12*, 4183. https://doi.org/10.3390/app12094183.

[21]. Lăcătușu, F.; Ionita, A.D.; Lăcătușu, M.; Olteanu, A. Performance Evaluation of Information Gathering from Edge Devices in a Complex of Smart Buildings. *Sensors* **2022**, *22*, 1002. https://doi.org/10.3390/s22031002

[22]. Dynatrace documentation. Available online: https://www.dynatrace.com/platform/ (accessed September 2021)

[23]. AppDynamics documentation. Available online: https://www.appdynamics.com/product/infrastructure-monitoring/cloud-monitoring (accessed September 2021)

[24]. NewRelic documentation. Available Online: https://docs.newrelic.com/docs/new-relic-solutions/get-started/intro-new-relic (accessed September 2021)

[25]. Olteanu, Adriana**;** Lacatusu, Marian; Ionita, Anca Daniela; Lacatusu, Florin, " Platform for Informal Education and Social Networking to Increase Awareness Regarding Nuclear Vulnerabilities" - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 333-340

[26]. Ioan Damian, Marian Lacatusu, Florin Lacatusu, Anca Daniela Ionita, Web Services for Guiding Persons with Locomotor Impairments in Public Spaces, *2022 26th International Conference on System Theory, Control and Computing* (ICSTCC), 2022, pp.639-644.

[27]. Adriana Olteanu, Florin Lacatusu, Marian Lacatusu, Iulian Craciun, Anca Daniela Ionita, "Mobile Application for Crisis Situations in a University Campus" - The

International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 280-287

[28]. M. Lacatusu, A. D. Ionita, F. Lacatusu and I. Damian, "Decision support for multicloud deployment of a modeling environment," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 247-251, doi: 10.1109/ICCP56966.2022.10053951.

[29]. Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN COMPUT. SCI.* **2**, 160 (2021). https://doi.org/10.1007/s42979-021-00592-x

[30]. H. Yang and Y. Kim, " Design and Implementation of Fast Fault Detection in Cloud Infrastructure for Containerized IoT Services," *Sensors,* pp. 20, 4592, 2020.

[31]. F. Lacatusu, A. D. Ionita, M. Lacatusu, I. Damian and D. Saru, "A Comparison of Cloud Edge Monitoring Solutions for a University Building," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 253-257, doi: 10.1109/ICCP56966.2022.10053978.

[32]. F. Lacatusu, I. Damian, A. D. Ionita and M. Lacatusu, Smart Building Manager Software in Cloud, *University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 83, no. 4,* 2021.

[33]. Kubernetes requests and limits. Available online:https://kubernetes.io/docs/concepts/configuration/manage-resources containers/ (accessed on August 16, 2021).

[34]. Marian Lacatusu, Florin Lacatusu, Ioan Damian, Anca Daniela Ionita, Multicloud Deployment to Support Remote Learning, 15th International Technology, Education and Development Conference (INTED 2021) Proceedings (2021), pp. 4601-4606.

[35]. Ioan Damian, Marian Lacatusu, Anca Daniela Ionita, Florin Lacatusu, Software Services to Support Faculty Management in Times of Pandemic, 15th International Technology, Education and Development Conference (INTED 2021) Proceedings (2021), pp. 4634-4639, IATED Digital Library, doi: 10.21125/inted.2021.0943.