Program PARTENERIATE - Proiecte colaborative de cercetare aplicativă - Competiţie 2013

RST - Raport ştiinţific şi tehnic

CALCULOS - Arhitectură cloud pentru o bibliotecă deschisă de blocuri funcţionale logice reutilizabile pentru sisteme optimizate

Codul proiectului: PN-II-PT-PCCA-2013-4-2123
Contract Nr.: 257/2014

Etapa II Arhitectura de sistem. Algoritmi pentru calculul riscului şi de detecţie a avariilor, diagnoză şi acomodare

Cuprins

1 Introducere

Obiectivul principal al proiectului este proiectarea unei platforme cloud şi a serviciilor asociate, platformă care va furniza resursele de prelucrare pentru accesarea şi rularea algoritmilor de control avansat şi optimizare a instalaţiilor industriale la scară mare. Aceste servicii vor permite utilizatorului să efectueze analize de risc online şi prevenirea pericolelor folosind algoritmi generici de control, optimizare, diagnoză, prevenire a avariilor şi analiza defecţiunilor.

Obiectivele specifice ale proiectului sunt:

- Proiectarea unei platforme care să folosească o interfaţă prietenoasă de programare adresată mai multor tipuri de utilizator;
- Crearea unor maşini virtuale care să poată găzdui module şi aplicaţii complexe;
- Reducerea costurilor de mentenanţă pentru instalaţiile industriale;
- Îmbunătăţirea relaţiei între mediul academic şi cel industrial;
- Îmbunătaţirea proceselor şi instalaţiilor industriale prin metode şi servicii accesibile.

În cadrul primei etape de execuţie a proiectului, *Etapa I — Cerinţele utilizatorului, analiza acceptabilităţii şi platforma colaborativă*, au fost efectuate de către parteneri patru activităţi principale, având ca rezultate un studiu privind stadiul industriei, cerinţele şi aşteptările sale, un raport de analiză ştiinţifică şi tehnică cu privire la posibilităţile de implementare a sistemului, elaborarea unei arhitecturi de control a proceselor şi, respectiv, analiza posibilităţilor de interfaţare cu utilizatorii şi cu procesul.

Obiectivul etapei de execuţie. Conform planului de realizare a proiectului, în cadrul acestei etape de execuţie a proiectului, *Etapa II — Arhitectura de sistem. Algoritmi pentru calculul riscului şi de detecţie a avariilor, diagnoză şi acomodare*, au fost efectuate de către parteneri următoarele activităţi principale:

- Activitatea II.1 (A2.1): Elaborarea documentaţiei pentru modelul funcţional,
- Activitatea II.2 (A2.2): Elaborare specificaţii tehnice pentru componentele sistemului,
- Activitatea II.3 (A2.3): Proiectare de algoritmi pentru calculul riscului,
- Activitatea II.4 (A2.4): Algoritmi de detecţie a avariilor, diagnoză şi acomodare.

Obiectivele propuse pentru această etapă a proiectului au fost atinse în întregime. Toate cele patru activităţi prevăzute au fost efectuate şi au fost orientate spre îndeplinirea obiectivului principal şi a obiectivelor specifice menţionate mai sus.

Implementarea aplicaţiilor de conducere automată avansată a proceselor industriale necesită uzual resurse sporite de stocare şi execuţie. Raportul prezintă o aplicaţie bazată pe cloud care îi permite unui utilizator să aibă acces la diferite strategii de conducere folosind o interfaţă web, să beneficieze de regulatoare virtualizate, care să execute acei algoritmi şi să trimită unele comenzi dispozitivelor din proces. Raportul propune o soluţie pentru interconectarea modulelor implementând noi servicii, utilizând interfeţe RESTful. Abordarea inovativă prezentată presupune virtualizarea unor "baze de date" de regulatoare de proces şi îi conferă inginerului automatist dintr-o întreprindere industrială accesul la strategii avansate de modelare, optimizare şi conducere, implementate ca funcţii bloc IEC 61499.

Raportul investigează, de asemenea, aspecte referitoare la managementul situaţiilor de risc, cât şi pentru detecţia avariilor, diagnoză şi acomodare.

Etapa II nu a presupus diseminarea explicită a unor rezultate. Totuşi, au fost publicate cinci lucrări elaborate de unii membri ai echipei proiectului (una dintre ele cu alţi doi colaboratori, care nu

fac parte din echipă), iar o altă lucrare va fi prezentată la o conferinţă care va avea loc la Budapesta în luna decembrie 2015 şi va fi publicată de către WSEAS (World Scientific and Engineering Academy and Society). Lucrările publicate deja au fost prezentate la conferinţe internaţionale co-sponsorizate de IEEE (Institute of Electrical and Electronics Engineers) şi sunt incluse în reputata colecţie IEEE Xplore Digital Library. Toate cele şase lucrări menţionează într-o secţiune de mulţumiri suportul financiar parţial din proiectul CALCULOS.

2 Elaborarea documentaţiei pentru modelul funcţional

Raportul prezintă un model pentru integrarea unei biblioteci de conducere avansată a proceselor într-un mediu bazat pe cloud. Scopul este de a reduce durata de elaborare a sistemelor automate, a efortului de întreţinere şi a complexităţii aplicaţiilor de conducere a proceselor, prin adoptarea unei abordări modulare constând din strategii de comandă generice, reutilizabile, conforme unei reprezentări din standardul IEC 61499. Soluţia utilizează tehnologii cloud deschise, cum ar fi Docker.

2.1 Oportunitatea utilizării platformelor cloud în conducerea proceselor

Evoluţia rapidă a platformelor de calcul cloud crează oportunităţi deosebite pentru întreprinderile industriale, din ce în ce mai complexe, pentru a realiza conducerea mai eficientă a proceselor şi reducerea costurilor operaţionale. De pildă, adoptarea soluţiilor cloud în sistemele de fabricaţie permite integrarea pe verticală a diferitelor niveluri în procesul de producţie, astfel încât conducerea proceselor poate fi concepută atât la nivelul supervizor (MES — Manufacturing Execution System), cât şi la nivelul planificării întreprinderii (ERP — Enterprise Resource Planning), asigurând o performanţă îmbunătăţită prin reducerea costurilor, creşterea flexibilităţii sistemului, a toleranţei la avarii, scalabilităţii şi agilităţii.

Calculul în cloud este o paradigmă emergentă care oferă atât echipamente de calcul cât şi programe sub formă de servicii. Paradigma posedă multe elemente atractive pentru utilizatori, dar şi pentru furnizorii de infrastructură şi de servicii ca, de exemplu, elasticitatea resurselor; se crează oportunităţi pentru reducerea energei, pe baza virtualizării şi a echilibrării sarcinilor de prelucrare [46]. Există studii despre provocările pe care caracteristicile fiecărui tip de aplicaţii le impune acestui mediu de execuţie [19]. În mediile industriale apar provocări speciale. Sistemele industriale actuale de conducere automată memorează mari cantităţi de date de la senzori de supraveghere, cu scopul de a optimiza procesele. În ultimele decade, au fost proiectate astfel de sisteme în principal în ipoteza că acestea operează în infrastructuri de tehnologie a informaţiilor (IT) închise, la nivelul întreprinderii, fără scalabilitate orizontală. Tehnologiile cloud pot fi utilizate în acest context pentru a reduce costurile IT locale şi a permite o scalabilitate sporită; totuşi, maturitatea acestor sisteme pentru aplicaţiile industriale, care au cerinţe severe de robusteţe şi viteză de răspuns (în timp real), nu este încă bine înţeleasă.

În cazul utilizării reţelelor fără fir (Wireless Area Networks—WANs) ca infrastructură de comunicaţii, latenţa şi lăţimea de bandă limitează utilitatea mediului de execuţie în cloud ca suport pentru aplicaţiile în timp real. Aceeaşi problemă apare când se utilizează Internetul pentru conducerea în reţea; latenţa Internetului este prea mare pentru funcţionarea în timp real a aplicaţiilor şi nu este posibil întotdeauna să se folosească un client hardware pentru a furniza capacitatea cerută.

Un suport important pentru a depăşi aceste dificultăţi este oferit de adoptarea SOA (Service Oriented Architecture) ca soluţie pentru nivelul superior (de deasupra MES), care conţine principalele puncte de integrare cu alţi furnizori de materiale şi servicii. Folosind SOA, sistemele cloud pot controla şi optimiza automat utilizarea resurselor prin oferirea unei capacităţi de contorizare la

un anumit nivel de abstractizare, corespunzător tipului serviciului (de pildă, memorare, prelucrare, lăţimea de bandă şi conturile utilizatorilor activi).

Principalul avantaj al folosirii prelucrării în cloud pentru conducerea proceselor constă în conversia sistemelor de calcul tradiţionale la mediile cu virtualizare. Virtualizarea unui sistem de calcul înseamnă organizarea şi gestionarea în comun a resurselor hardware şi software, permiţând mai buna lor folosire, auto-servire fără cerere şi elasticitate rapidă a capacităţilor de furnizare de resurse. Importanţa acestor elemente a generat eforturi de cercetare susţinute. Drept urmare, au fost propuse şi implementate (cu finalizare între anii 2013 şi 2015), câteva proiecte în cadrul programului FP7 dintre care se menţionează 4CAAST, CONTRAIL, LEADS şi VISION Cloud.

Abordarea din acest proiect şi din lucrările elaborate de echipă [4, 5, 6, 16, 17] este de a furniza puterea de calcul aproape de client, acceptând latenţe scăzute, fără a mări cerinţele de capacitate de calcul pentru clienţi. Această abordare de proiectare permite realizarea unor sisteme eficiente de conducere automată bazate pe Internet, prin extinderea reţelelor SCADA locale de la nivelul supervizor, capabile să implementeze strategii de conducere reutilizabile urmărind o reprezentare IEC 61499 [29]. Aceste strategii, cât şi rezultatele execuţiei lor, sunt făcute disponibile ca servicii cloud. Formatul de date recomandat pentru acest scop este protocolul OPC, care oferă o interfaţă de comunicaţie unificată pentru un număr mare de tipuri de echipamente şi modele. Rezultatul principal este un model pentru integrarea unei biblioteci avansate de conducere a proceselor într-un mediu bazat pe cloud.

2.2 Utilizarea specifică a serviciilor cloud în conducerea proceselor

Standardul IEC 61499 a fost elaborat ca soluţie de mare performanţă pentru a permite automatizarea inteligentă încorporată în componente software cărora li se asociază funcţii bloc (FB), la rândul lor distribuite în sisteme cu reţele [43]. Funcţiile bloc IEC 61499 sunt proiectate natural drept componente reutilizabile, distribuite de la nivelul de conducere inferior până la nivelul cel mai înalt. Abordarea din proiect este de a considera funcţiile realizate prin FB disponibile ca servicii cloud. Aşadar, la nivel logic, funcţionalitatea este încapsulată în servicii, care pot fi invocate de alte servicii pentru a realiza o sarcină, iar toate datele din proces sunt accesibile independent ca servicii.

Întrucât calculul cloud oferă diverse soluţii de punere la dispoziţie a unei infrastructuri dinamice şi flexibile de resurse de calcul, furnizate ca servicii la cerere, acesta poate fi considerat ca o soluţie promiţătoare pentru sistemele de automatizare industriale ale viitorului, care trebuie să fie adaptabile şi agile. Unul dintre cele mai promiţătoare modele arhitecturale este descris în [18]. Modelul constă din două etaje principale. Etajul inferior reprezintă migrarea nivelurilor din proces şi de automatizare, în timp ce etajul superior reprezintă nivelurile avansate de automatizare şi management. Este propus un cloud de automatizare (AT-Cloud), care furnizează funcţii şi servicii la nivelul inferior, şi aplicaţii şi servicii (IT-cloud) la nivelul superior. Integrarea între aceste două infrastructuri este asigurată de un Model Informaţional. Din acest model general de cloud pentru automatizare pot fi derivate atât modelele disponibile cât şi cele viitoare de calcul cloud. Pot fi evidenţiate următoarele modele de furnizare a serviciilor cloud: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) şi Software as a Service (SaaS), cât şi modelele de implementare (private, publice şi hibride). Funcţiile şi serviciile de automatizare speciale pot fi oferite direct ca aplicaţii SaaS din IT-cloud, cât şi din AT-cloud. Alternativ, o aplicaţie PaaS este utilizată ca platformă de automatizare pentru a furniza necesităţi specifice pentru integrare.

În continuare, se prezintă structura şi funcţiile unei noi arhitecturi conceptuale ierarhice, proiectate astfel încât să evidenţieze conceptul de servicii software auto-adaptive deschise, care pot fi evaluate într-un efort de a structura sistemele ICT emergente şi de servicii software distribuite, necesare în automatizarea proceselor industriale. Există patru provocări principale care au primit

soluţionări adecvate prin monitorizarea aplicaţiilor cloud utilizând arhitecturile propuse, fundamental diferite de monitorizarea tradiţională a aplicaţiilor în întreprinderi industriale.

i. **Virtualizarea**. Pe lângă simpla monitorizare a elementelor cheie ale sistemului la fiecare nod fizic individual, folosirea virtualizării crează o rezervă de capacitate dinamică de resurse de calcul şi de stocare, care trebuie monitorizate şi administrate într-un mod diferit.

ii. **Determinarea profilului timpilor de răspuns pentru utilizatorul final**. Monitorizarea performanţei utilizatorului este diferită într-o aplicaţie cloud, deoarece operează pe o reţea publică deschisă — fapt ce determină provocări proprii în ceea ce priveşte capacitatea de a monitoriza efectiv timpii de răspuns.

iii. **Testarea încărcării la scara web**. Soluţia propusă înclină în favoarea aplicaţiilor cloud a diferenţelor de scară, în competiţia dintre scara întreprinderii şi scara web. Scara întreprinderii este mărginită, predictibilă şi măsurabilă, în timp ce scara web, experimentată prin aplicaţiile cloud, nu este. Vârfurile de trafic într-o aplicaţie cloud sunt posibil nelimitate, dar doar aparent impredictibile, deoarece istoricul datelor procesului oferă o informaţie specială ca Data as a Service (DaaS), care poate fi folosită pentru elaborarea algoritmilor de predicţie bazaţi pe serii de timp.

iv. **Multi-închiriere**. Arhitecturile multi-închiriere furnizează o singură instanţă a unei aplicaţii, pe baza unui model comun de date servind multipli "chiriaşi". Această abstracţie logică face mai dificilă monitorizarea şi profilarea indicatorilor de performanţă ai unui client individual.

2.3 Descrierea arhitecturii sistemului

Proiectul urmează o abordare de conducere avansată a proceselor, care permite configurarea execuţiei funcţiilor bloc ale standardului IEC 61499 pe un server la distanţă utilizând o interfaţă web standard şi trimite rezultatele la regulatoarele procesului industrial folosind interfeţe de proces specifice [33]. Se doreşte creşterea eficienţei execuţiei şi a flexibilităţii, prin aplicarea aceloraşi concepte într-o structură scalabilă dinamică, cu resurse extinse. Arhitectura sistemului a fost proiectată cu scopul de a simplifica migrarea de la conducerea centralizată la un model distribuit, în care fiecare algoritm de prelucrare poate fi executat pe o maşină diferită. Acest model poate fi implementat folosind o platformă cloud PaaS.

Componentele sistemului

Sistemul este proiectat ca o aplicaţie bazată pe web, care să permită inginerilor automatişti din industrie să acceseze şi să execute diferiţi algoritmi complecşi sau diferite strategii (reglare multivariabilă, conducere adaptivă, conducere predictivă, identificarea sistemelor etc.), pe baza datelor memorate din proces şi să optimizeze parametrii procesului pe baza acestor rezultate. Aplicaţia web este editată pe o platformă cloud, unde sunt definite două servicii de stocare, Data as a Service şi Algorithm as a Service, cât şi două servicii de execuţie, Model as a Service şi Control as a Service.

Data as a Service (DaaS) permite unui utilizator să aibă acces la cloud pentru date istorice, dar cu securitate, scalabilitate şi fiabilitate sporite, cât şi cu funcţionalităţi de toleranţă la defecte sau avarii. Culegerea datelor poate fi făcută fie dintr-o bază de date de proces existentă, fie dintr-o aplicaţie de istoric SCADA, fie în timp real folosind o interfaţă OPC-UA cu regulatoarele din proces sau noduri cu senzori inteligenţi. Algorithm as a Service (AaaS) furnizează o reprezentare IEC 61499 a algoritmilor disponibili şi a strategiilor disponibile, care pot fi ajustate şi utilizate în logica existentă a unui regulator de proces. Toţi algoritmii sunt furnizaţi într-o structură generică, uşor de adaptat la procese specifice. Model as a Service (MaaS) poate fi utilizat pentru a executa identificarea sistemului şi optimizarea parametrică a procesului folosind datele stocate. Control as a Service (CaaS) a fost proiectat pentru a permite execuţia de la distanţă a algoritmilor disponibili,

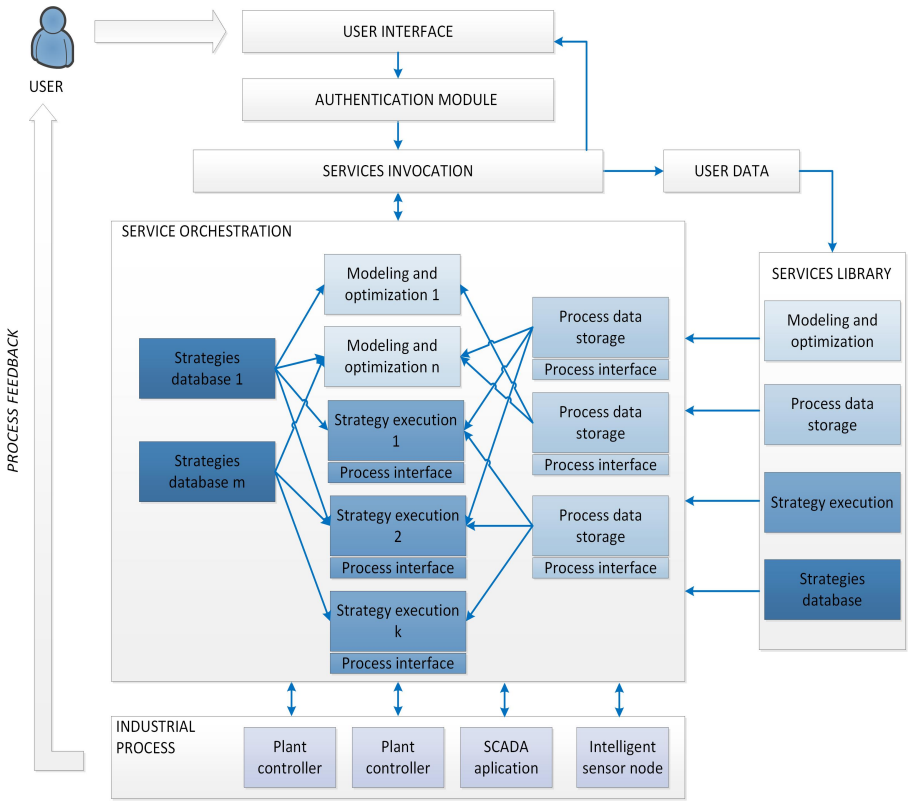

Figura 1: Arhitectura sistemului.

utilizând date de timp real din proces. Datorită naturii impredictibile a resurselor cloud şi a vitezei comunicaţiilor, acest serviciu nu este adecvat aplicaţiilor critice, dar oferă funcţionalitate importantă pentru aplicaţii cum ar fi ajustarea modelului pe baza datelor de timp real din proces, analiza seturilor mari de date din zone larg dispersate geografic etc.

Arhitectura sistemului este prezentată în Fig. 1 [6]. Un utilizator poate accesa interfaţa web pentru a solicita unul dintre serviciile disponibile. Modulul de Autentificare restricţionează accesul unui utilizator doar la informaţiile generale şi informaţia sa personală. Serviciile disponibile sunt organizate în patru module: modulul de Modelare şi Optimizare oferă MaaS, modulul de Stocare a Datelor din Proces rezolvă cererile DaaS, modulul de Execuţie a Strategiilor implementează CaaS, iar modulul Baza de Date de Strategii este responsabil pentru AaaS. Toate aceste module, preconfigurate cu funcţionalitatea dorită, sunt disponibile într-o bibliotecă a sistemului şi pot fi instanţiate pe baza datelor utilizatorului când se primeşte o cerere de serviciu.

Platforma cloud este responsabilă pentru orchestrarea serviciilor cererilor active: alocarea dinamică a resurselor virtuale, cât şi lansarea în execuţie şi terminarea aplicaţiilor cloud. Pot fi interconectate diferite instanţe, de exemplu, dacă este executată o strategie folosind datele din proces din câteva resurse, sau dacă sunt utilizate simultan diferite instanţe ale aceleiaşi strategii. Interconectarea cu procesul se realizează folosind interfeţe de proces create dinamic pe baza informaţiilor furnizate de utilizator privind parametrii de comunicaţie. În acest mod pot fi utilizate în diferite aplicaţii strategiile generice disponibile.

Tehnologii utilizate

Alegerea tehnologiilor utilizate pentru elaborarea sistemului a luat în considerare accesul la funcţionalitatea dorită, uşurinţa implementării şi instrumentele de dezvoltare a aplicaţiilor centralizate. S-a ales o reprezentare a algoritmilor bazată pe funcţiile bloc ale standardului IEC 61499, datorită caracterului lor deschis, portabilităţii, independenţei de platformă şi posibilităţii de a proiecta arhitecturi distribuite. Regulatoarele locale dintr-un proces industrial pot utiliza resurse cloud prin conectarea la regulatoarele virtuale definite în modulul de Execuţie a Strategiilor.

Ca mediu de execuţie runtime a funcţiilor bloc IEC 61499 s-a ales instrumentul gratuit FBRT, bazat pe Java. Avantajele sale principale sunt compatibilitatea completă cu standardul IEC 61499 şi cu alte produse, cât şi costul şi flexibilitatea oferite pentru rularea sau oprirea aplicaţiilor. Pentru a face rezultatele execuţiei disponibile ca servicii, permiţând astfel comunicaţia inter-module în cloud, s-au definit interfeţe RESTful care folosesc servicii web peste HTTP pentru transferul datelor.

La selectarea celei mai bune infrastructuri cloud pentru aplicaţii de conducere automată avansată s-au luat în considerare disponibilitatea alocării dinamice a resurselor de calcul, fiabilitatea datelor stocate, scalabilitatea şi accesul la cerere. Toate acestea pot fi obţinute cel mai bine, în această aplicaţie, utilizând o infrastructură PaaS. Sunt disponibile câteva soluţii PaaS, cum ar fi Amazon Web Services (AWS), Microsoft Azure şi Google Cloud Platform. Fiecare dintre acestea oferă soluţii diferite cu privire la tehnologiile colaterale pentru server, suport pentru instrumentele de dezvoltare şi de integrare a aplicaţiilor.

Heroku este una dintre cele mai de succes soluţii PaaS create peste Amazon EC2, care permite punerea în funcţiune, rularea şi administrarea aplicaţiilor scrise în Ruby, Node.js, Java, Python, Clojure, Scala şi PHP. Chiar dacă Heroku aduce multe beneficii în privinţa uşurinţei de utilizare, dependenţa sa de platforma Amazon EC2 nu permite flexibilitatea şi deschiderea aşteptate pentru acest sistem de dezvoltare. De asemenea, Heroku nu oferă caracteristici de toleranţă la defecte, scalabilitate automată sau fiabilitatea datelor, necesare pentru această aplicaţie.

Docker este o tehnologie deschisă (open source) care oferă o virtualizare la nivelul sistemului de operare, similară maşinilor virtuale, dar cu "balast" semnificativ mai redus. Aceeaşi aplicaţie poate fi descărcată pe platforme diferite, decuplându-se astfel cerinţele de infrastructură de mediul aplicaţiei. Portabilitatea Docker permite ca utilizatorul să îl integreze în toate soluţiile IaaS/PaaS publice disponibile, cât şi în infrastructuri private. În plus, Docker face posibilă automatizarea procesului de includere a unei aplicaţii în containere şi oferă o interfaţă de înalt nivel pentru administrarea containerelor. Datorită performanţei sale, consumului redus de resurse, simplităţii de împachetare şi depunere a unei aplicaţii în containere, cât şi largului suport din industrie, Docker a devenit un standard popular pentru implementarea aplicaţiilor în cloud, în special pentru proiectele la care portabilitatea este esenţială.

2.4 Detalii de implementare

Sistemul cuprinde patru module de bază care sunt instanţiate în funcţie de numărul de utilizatori şi de serviciile şi aplicaţiile cerute de aceştia. Se va crea o instanţă a fiecărui modul pentru fiecare cerere de serviciu solicitat de un nou utilizator sau o nouă aplicaţie. Componentele fiecărui modul, prelucrarea datelor şi interacţiunea cu alte componente ale sistemului sunt prezentate în Fig. 2 [6]. Fluxul de date CaaS între componentele modulului de Execuţie a Strategiilor este ilustrat cu roşu. Partea verde reprezintă administrarea cererilor DaaS utilizând componentele modulului de Stocare a Datelor din Proces. Liniile albastre conectează modulul de Modelare şi Optimizare la alte componente ale sistemului. Liniile purpurii furnizează accesul la modulul Bazei de Strategii.

Utilizatorii pot accesa serviciile disponibile utilizând o interfaţă web. Când un utilizator emite

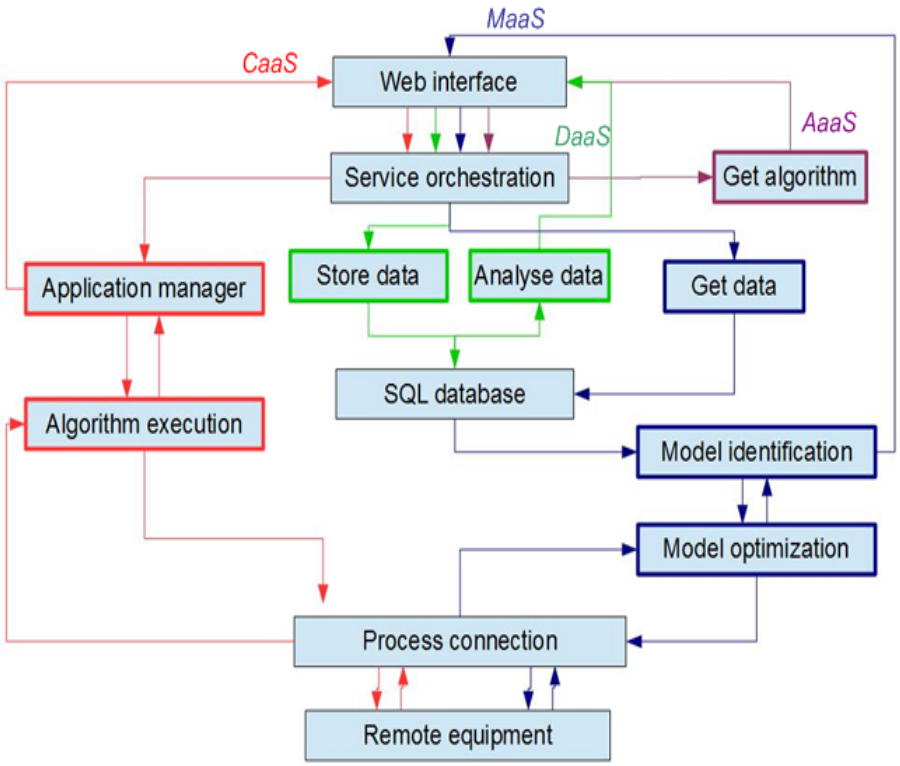

Figura 2: Interconectarea modulelor sistemului.

o cerere de serviciu CaaS, trebuie să selecteze un algoritm specific şi să precizeze parametrii de execuţie şi detaliile de comunicaţie ale echipamentului de la distanţă. Serverul web primeşte cererea, extrage informaţia privind detaliile de execuţie (secvenţa ID, comunicaţia cu dispozitivul de la distanţă, parametrii algoritmului de executat) şi crează o nouă instanţă a execuţiei containerului.

Un utilizator poate accesa serviciul DaaS ca un "istoric" cloud, permiţându-i să memoreze datele din proces pe baza unei adrese specificate. Datele pot fi extrase dintr-un server SQL existent, sau pot fi obţinute dintr-o aplicaţie SCADA folosind OPC. Se poate efectua analiza datelor, se pot determina tendinţe, sau genera diferite rapoarte, integrarea în cloud asigurând reducerea timpilor de prelucrare şi fiabilitatea sporită a spaţiului de stocare.

Echipamentul la distanţă poate fi reprezentat de un regulator de proces, o aplicaţie SCADA, sau chiar de senzori inteligenţi. În primele două cazuri, comunicaţia poate fi făcută folosind o interfaţă OPC cu procesul. În ultimul caz se preferă o soluţie mai eficientă, de pildă un mecanism publicare-abonament implementat în IEC 61499, care va permite, de exemplu, transmiterea datelor doar când apar schimbări, reducând astfel consumul de lăţime de bandă.

3 Specificaţii tehnice pentru componentele sistemului

Pentru a realiza implementarea proiectului este necesar un mediu SOA contruit pe servicii web care să furnizeze o interfaţă pentru dispozitive IEC 61499 şi sisteme distribuite de conducere automată. Aceste sisteme automate pot fi abstractizate prin dispozitive virtuale şi interfaţate cu o interfaţă REST cu scopul de a furniza blocurile constructive de bază pentru servicii şi algoritmi de conducere complecşi. În acest mod, pot fi proiectate structuri de conducere automată de înalt nivel, pe baza

unor servicii standard, flexibile, care să poată fi implementate şi încorporate în cloud.

Un model conceptual pentru serviciul web de bază pentru conducere automată este prezentat în Fig. 3 din [6]. Sistemele de conducere automată IEC 61499 se bazează pe funcţii bloc (FB), care încapsulează date şi algoritmi, într-o unitate abstractă de conducere. În plus, mai multe funcţii bloc pot fi grupate împreună pentru a crea reţele FB. Aceste funcţii bloc sunt asociate cu dispozitive automate, şi pot fi încorporate în orice resursă care poate găzdui sistemul "runtime" IEC 61499. De exemplu, trusa runtime şi de dezvoltare pentru funcţii block, FBDK2, foarte populară şi bazată pe Java, poate fi încărcată pe orice sistem care suportă JVM (Java virtual machine).

3.1 Structura algoritmilor

Algoritmii vor fi reprezentaţi sub forma unor funcţii bloc compuse, cu o structură deschisă, astfel încât utilizatorul să poată efectua modificări în cazul în care doreşte acest lucru. Algoritmii vor fi disponibili sub forma unor fişiere de tip `*.fbt`, compatibile cu toate mediile de dezvoltare care implementează standardul IEC 61499. Fiecare algoritm va fi însoţit de o scurtă descriere care să detalieze funcţionalitatea îndeplinită de acesta, dacă este aplicabil numai unor anumite procese, domeniul de utilizare, denumirea şi rolul parametrilor de intrare şi de ieşire (date şi evenimente) şi eventual rezultatul aşteptat în urma implementării. Algoritmii care implementează funcţii mai complexe (de modelare, optimizare, analiză etc.) vor putea fi executaţi în bibliotecă pentru ca apoi rezultatul să fie trimis către un echipament aflat la distanţă. Pentru ca acest lucru să fie posibil, funcţia bloc compusă specifică algoritmului va fi introdusă într-o structură de tip configuraţie de sistem cu extensia `*.sys` având definit un dispozitiv (de tip `PANEL_DEVICE`) şi o resursă (de tip `PANEL_RESOURCE`). Aceasta este configuraţia minimă care permite simularea unui sistem distribuit şi oferă în plus facilităţi de operare specifice echipamentelor industriale, precum pornire la cald, pornire la rece sau oprire.

Blocurile ce trebuie adăugate obligatoriu sunt cele care asigură controlul execuţiei (prin blocurile `RUNSTOP` şi `RS_GATE`) şi simulează ceasul unui echipament (prin blocul de tip `_CYCLE`), pornind de la o perioadă de eşantionare stabilită, precum şi cele care asigură interfaţarea cu aplicaţia web şi cu procesul. Pot fi necesare blocuri de preluare a parametrilor de intrare şi de ieşire din algoritm, pentru urmărirea mai uşoară a execuţiei algoritmilor şi recepţia corecte a parametrilor de la pagina web din cadrul serverului de aplicaţie.

Pentru integrarea în sistemul de execuţie online al bibliotecii, funcţiei bloc compuse i se adăugă un bloc de `Subscribe` conectat la parametrii de intrare ai funcţiei bloc, care permite preluarea parametrilor de execuţie de la interfaţa web, şi un bloc de `Publish` conectat la parametrii de ieşire ai funcţiei bloc, care asigură transmiterea rezultatului către echipamentul aflat la distanţă în instalaţie. Sunt cazuri în care algoritmii se execută în funcţie de unul sau mai mulţi parametri de proces. Pentru aceasta, este necesară adăugarea unui bloc de `Subscribe` suplimentar pentru conectarea la acei parametri.

Pentru ca procesul de generare al IP-urilor de la interfaţa web să se poată face automat, blocurile de comunicaţie `Publish-Subscribe` vor avea parametrul `ID` sub forma unui identificator generic, ulterior înlocuit cu valoarea adresei de IP şi a portului. Astfel, blocul de `Subscribe` care preia datele de la interfaţa web va avea ca identificator stringul `__CALCULOS_TO_FBDK__`, blocul de `Publish` care transmite rezultatul către instalaţie va avea identificatorul `__FBDK_TO_PLANT__`, iar în eventualitatea că există un bloc de `Subscribe` ce preia din instalaţie valorile parametrilor de proces, acesta va avea identificatorul `__PLANT_TO_FBDK__`.

3.2 Model conceptual al sistemului distribuit de control cu componente în cloud

În continuare, se prezintă detaliat un model conceptual pentru un sistem de control distribuit bazat pe utilizarea funcţiilor bloc ale standardului IEC 61499, în care anumite dispozitive sunt reprezentate de aplicaţii care se execută în cadrul unei infrastructuri cloud. Acestea sunt incluse în containere Docker ce pot fi instanţiate la majoritatea furnizorilor de servicii cloud publice, precum şi în sisteme de calcul locale sau private, fiind o soluţie standard pentru distribuirea aplicaţiilor.

Sistemul distribuit de control are o arhitectură client-sever. Componenta server implementează funcţia bloc server sau o reţea de funcţii bloc în care este inclusă funcţia bloc de tip server. Componenta client implementează funcţia bloc client sau o reţea de funcţii bloc în care este inclusă funcţia bloc de tip client. Aceste funcţii bloc, respectiv client şi server, oferă facilitatea de comunicare utilizând protocoalele TCP/IP. Pentru implementarea acestei arhitecturi se propune utilizarea mediului de execuţie FBDK2. Acesta este implementat în limbajul Java, astfel încât funcţiile bloc pot fi incluse în cadrul unor programe complexe care pot conţine elemente de interfaţă cu utilizatorul sau apeluri de funcţii externe pentru execuţia unor algoritmi complecşi care sunt implementaţi în alte limbaje de programare. De asemenea, pentru a putea compune aplicaţii care utilizează paradigma SOA, se propune proiectarea unei interfeţe de tip REST, astfel încât componenta client a sistemului distribuit de control să poată fi accesată sub forma unor servicii web de tip RESTful.

În această configuraţie, atât componenta client cât şi componenta server sunt implementate în cadrul unor containere Docker diferite. Acestea pot fi instanţiate în mod separat, neexistând nici o legătură între aceste sisteme din punct de vedere al operării serviciilor. Astfel, componenta client poate să se execute în cadrul unui serviciu cloud public, iar componenta server se poate executa în cadrul unei infrastructuri cloud private. Singura condiţie pentru funcţionarea sistemului o reprezintă comunicarea directă utilizând protocolul TCP.

Containerele Docker permit execuţia mai multor procese, însă cele mai bune practici recomandă execuţia doar a unui singur proces în cadrul unui container, şi compunerea unor stive de containere pentru executarea unor aplicaţii care necesită prezenţa unui server web, a unei baze de date, sau a altor procese auxiliare. O soluţie în acest sens este oferită de platforma Kubernetes ce este dezvoltată de Google [25]. Astfel, mai multe containere care furnizează componente necesare unui serviciu sau unei aplicaţii sunt grupate împreună şi formează o entitate denumită *pod*, ce este instanţiată de către platforma care administrează resursele [31]. Această configuraţie este prezentată în Fig. 3. În acest exemplu, sistemul distribuit de control este executat în cloud, iar interfaţa sub forma serviciilor web reprezintă o soluţie pentru crearea unui model de aplicaţie de tip CaaS (Control as a Service).

3.3 Experimentarea modelului client-server al sistemului de control IEC 61499

Pentru experimentarea acestui model al sistemului de control, s-a utilizat un exemplu propus în cadrul grupului de discuţii al platformei FBDK2, respectiv un sistem de control care avea funcţia de server şi un sistem care îndeplinea funcţionalitatea client [15]. Sistemul server a fost instalat pe o maşină virtuală găzduită în cadrul serviciului cloud public Microsoft Azure. Adresa serverului este: `calculos.cloudapp.net`. Pentru conectare trebuie utilizat portul `4444/tcp`. Serverul execută o operaţie extrem de simplă, respectiv calculează suma a două numere şi afişează rezultatul obţinut. Pentru execuţia clientului s-a folosit comanda: `java -jar demo.jar`. Fereastra client este prezentată în Fig. 4.

Un al doilea experiment a inclus utilizarea unui container Docker şi implementarea serverului în cadrul acestuia. Pentru lansarea în execuţie a serverului trebuie utilizat containerul Docker denumit `alexcstanciu/calculos_demo2`. Spre exemplu, se utilizează următoarea procedură:

1. Se descarcă imaginea `alexcstanciu/calculos_demo2`

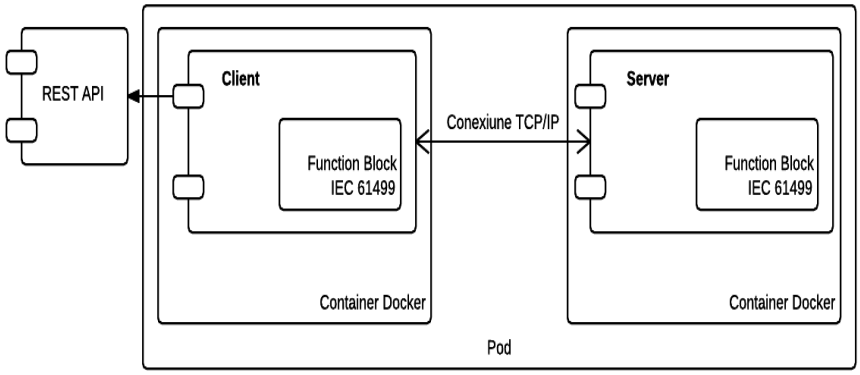

Figura 3: Model conceptual de containere Docker compuse într-o structură comună — pod.

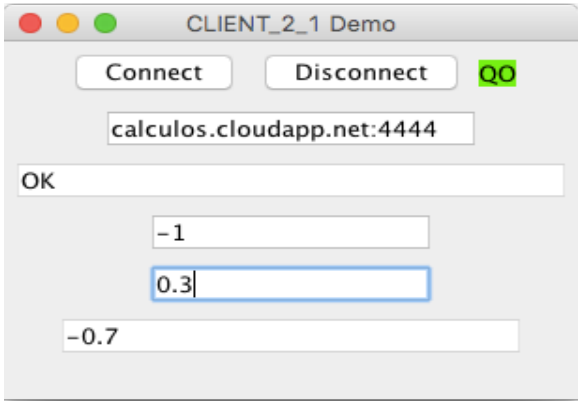

Figura 4: Implementarea unui server în cadrul serviciului cloud Microsoft Azure.

```
$ docker pull alexcstanciu/calculos_demo2
Using default tag: latest
latest: Pulling from alexcstanciu/calculos_demo2
d29d374e51d0: Pull complete
...
Digest: sha256:9797fd50221b74ba93752422291386bd9bccf86a9606384c0cc518226838e27b
Status: Downloaded newer image for alexcstanciu/calculos_demo2:latest
```

2. Se instanţiază un container cu această imagine:

```
$ docker run -d -P alexcstanciu/calculos_demo2
23c04aacb423d34cbc35c977702e8875af2b7e5c2876161c8efbf44197aba97d
```

3. Pentru conectarea clientului avem nevoie de adresa ip a maşinii virtuale unde rulează Docker, precum şi de portul unde este accesibilă aplicaţia.

```
$ docker ps
CONTAINER ID  IMAGE                          COMMAND            CREATED
23c04aacb423  alexcstanciu/calculos_demo2  "/bin/sh -c demo"  16 seconds ago
STATUS         PORTS                     NAMES
Up 16 seconds  0.0.0.0:32768->4444/tcp  goofy_pasteur
```

```
$ docker-machine ip default
192.168.99.100
```

Fereastra client de similară celei din Fig. 4. (Adresa serverului şi portul sunt înlocuite cu numarul ip `192.168.99.100`, respectiv `32768`.

Pentru implementarea serviciului web de tip RESTful în cadrul componentei client trebuie avut în vedere modul în care se va face transferul datelor, respectiv alegerea unui format pentru serializarea datelor. Cele mai populare standarde utilizate în acest sens sunt JSON şi YAML. În continuare se va face o scurtă prezentare a acestora.

3.4 Serializarea datelor

Serializarea este procesul prin care structuri de date sau obiecte sunt transformate într-un format care poate fi stocat (într-un fişier sau într-un spaţiu de memorie) sau poate fi transmis în reţea şi care permite apoi reconstruirea datelor în starea iniţială pe un alt sistem de calcul [35]. Acest proces se utilizează pentru transferul datelor pe canale de comunicaţii, pentru salvarea datelor în baze de date sau pe disc, pentru executarea unor apeluri de proceduri la distanţă sau apelarea de servicii, etc. Serializarea trebuie să fie independentă de arhitectura sistemului de calcul. Cele mai populare formate pentru serializarea datelor sunt XML, JSON şi YAML, acestea având calitatea de a putea fi citite de o persoană, cât şi Thrift, Avro, sau Protocol Buffers, care au o reprezentare binară.

Standardul JSON

JavaScript Object Notation (JSON) este un format de tip text pentru serializarea datelor structurate. Acesta este derivat din limbajul JavaScript aşa cum a fost specificat în cadrul standardului ECMAScript. JSON poate descrie patru tipuri de date fundamentale (şiruri de caractere, numere, boolean şi null) şi două tipuri de date structurate (obiecte şi vectori) [2].

Un şir de caractere reprezintă o secvenţă formată din zero sau mai multe caractere de tip Unicode. Un obiect "reprint" este o colecţie neordonată formată din zero sau mai multe perechi nume/valoare, unde numele este un şir de caractere, iar valoarea poate să fie de tipul şir de caractere, număr, boolean, null, obiect sau vector. Termenii de 'obiect' şi 'vector' provin din limbajul JavaScript. Un *vector* este o listă ordonată din zero sau mai multe valori, fiecare putând să aibă un alt tip de date. Vectorii utilizează notaţia cu paranteze pătrate şi elementele separate cu virgulă. Un *obiect* este o colecţie neordonată de perechi nume/valoare în care numele, care mai este denumit şi *cheie*, este un şir de caractere. Obiectele sunt utilizate pentru a reprezenta vectori asociativi şi de aceea se recomandă ca fiecare cheie să fie unică în cadrul unui obiect. Pentru delimitarea obiectelor se folosesc acoladele, iar pentru separarea perechilor cheie-valoare se foloseşte virgula.

Obiectivele declarate în proiectarea formatului JSON au fost ca acesta să fie minimal, portabil, textual şi să reprezinte un subset al limbajului JavaScript. JSON este un standard deschis bazat pe text, care poate fi citit de o persoană, şi care este utilizat pentru transmiterea datelor. Acesta este principalul format folosit în cadrul comunicării asincrone browser-server web (AJAJ), înlocuind formatul XML ce este utilizat în cadrul comunicării de tip AJAX. Deşi se bazează pe limbajul JavaScript, JSON este un format independent de limbaj, existând biblioteci şi instrumente pentru generarea şi extragerea de date în format JSON în majoritatea limbajelor de programare [24].

Schema unui document JSON reprezintă un format de tip JSON ce defineşte structura datelor JSON cu scopul de validare şi documentare. Schema unui document JSON este similară unui contract referitor la modul în care sunt descrise datele necesare unei aplicaţii precum şi la modul în care acestea pot fi modificate. Schema JSON se bazează pe concepte din cadrul XML Schema (XSD) însă utilizează elemente JSON. JSON reprezintă o alternativă pentru standardul XML, aceste două

formate de date fiind larg utilizate pentru crearea, citirea şi transmiterea datelor. În afară de XML, un alt format foarte popular este YAML.

Standardul YAML

YAML este un format pentru serializarea datelor ce poate fi înţeles uşor de către o persoană. Denumirea YAML provine de la "YAML Ain't Markup Languge", deşi în primele iteraţii în dezvoltare se numea "Yet Another Markup Language". Această schimbare de nume s-a făcut pentru a evidenţia orientarea sa pentru transmiterea datelor şi nu pentru marcarea textului în cadrul unui document [44].

Sintaxa YAML a fost proiectată pentru a putea fi mapată cu uşurinţă pe tipurile de date cele mai utilizate în cadrul limbajelor de nivel înalt, şi anume: listă, vector asociativ şi scalar. Având o structură generală bazată pe indentarea nivelurilor inferioare, acest format este adecvat pentru situaţii în care o persoană trebuie să creeze sau să editeze structuri de date precum fişiere de configurare sau antetul unui document. Un rol deosebit în accesibilitatea acestui format îl are renunţarea la formele de demarcare precum paranteze, acolade, ghilimele, ce sunt dificil de urmărit în cadrul unor ierarhii imbricate. În plus, utilizarea spaţiilor şi a capătului de rând ca delimitatori îl fac să fie uşor de integrat cu instrumente precum grep/Python/Perl/Ruby.

Atât JSON cât şi YAML au ca obiectiv să fie uşor de citit de către o persoană, însă au priorităţi diferite. Principalul scop în proiectarea formatului JSON a fost să fie foarte simplu şi universal. Din acest motiv, JSON este uşor de generat şi de prelucrat dar este un format mai greu de citit. De asemenea, JSON se bazează pe un model informaţional extrem de simplu, ceea ce îl face să fie procesat de toate mediile de programare moderne. Pe de altă parte, YAML a avut drept obiectiv principal lizibilitatea şi posibilitatea de a serializa structuri de date arbitrare. Din acest motiv, YAML permite crearea de documente care sunt uşor de citit, însă este mai dificil de generat şi de prelucrat. De asemenea, YAML utilizează modele mai complexe pentru tipurile de date, ceea ce conduce la creşterea efortului de procesare atunci când sunt medii de programare mixte.

YAML poate fi privit ca un superset natural al standardului JSON ce oferă o lizibilitate mai bună şi un model informaţional superior. Acest lucru este confirmat în practică deoarece fiecare fişier JSON este şi un document valid în formatul YAML. Din acest punct de vedere este uşor de migrat de la formatul JSON la YAML dacă sunt necesare facilităţi suplimentare [45].

4 Proiectare de algoritmi pentru calculul riscului

Teoria riscului este strâns legată de teoria deciziei în condiţii de incertitudine [34]. Dintre conceptele de bază, menţionăm: riscul, măsura pentru calculul riscului şi evaluarea riscului. Subiectul principal al teoriei riscului îl constituie diversele forme de risc suportate de către indivizi, firme, organisme guvernamentale şi alte organisme cu rol de decizie confruntate cu necesitatea de a alege între diferite acţiuni posibile în situaţii cu rezultate nesigure, atunci când acestea sunt guvernate de către mecanisme aleatoare neutre sau de către un adversar inteligent. Riscul există în toate formele activităţii umane, iar studiul teoretic al naturii riscului şi al strategiilor pentru realizarea unor compromisuri a devenit un subiect central în decizia politică, economie (la nivel microeconomic, dar şi macroeconomic), inginerie, managementul mediului, teoria asigurărilor şi multe alte domenii.

Noţiunea de risc implică aleatoriul, astfel încât instrumentul de bază al teoriei riscului este calculul probabilităţilor. Exemple de posibile riscuri sunt, de pildă, întreruperea sau funcţionarea necorespunzătoare a unui proces tehnologic datorită defectării unei instalaţii, a unor senzori sau echipamente, căderii tensiunii de alimentare, producerii unor dezastre etc.

Severitatea unui risc depinde atât de probabilitatea lui de apariţie, cât şi de impactul acestuia (adică, severitatea consecinţelor) [32]. Riscuri mici sunt acelea care au o probabilitate mică de

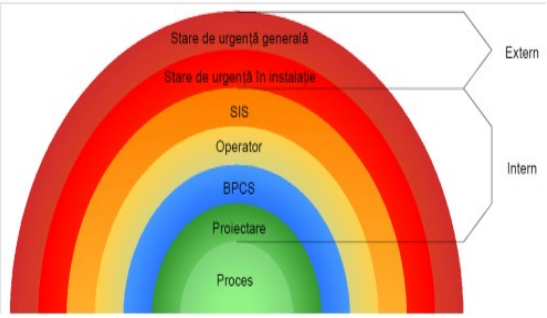

Figura 5: Niveluri de siguranţă pentru proces.

apariţie şi impact mic. Similar, sunt definite riscurile moderate sau mari. Un risc cu impact mic este considerat ca risc mic, chiar dacă probabilitatea sa de apariţie este mare. Severitatea poate fi apreciată în funcţie de produsul dintre probabilitate şi impact. În [32] sunt atribuite ponderile 1, 2 şi 3 riscurilor cu probabilitate mică, moderată şi, respectiv, mare şi ponderile 1, 3 şi 6 pentru impact mic, moderat şi, respectiv, mare, astfel încât ponderea globală a riscului, obţinută ca produs al ponderilor pentru probabilitate şi impact, are una dintre valorile 1, 2, 3, 6, 9, 12 şi 18. Astfel se pot identifica riscurile cele mai critice şi pot fi aplicate strategii sau politici de control adecvate.

Riscurile pot fi clasificate în mai multe categorii, dintre care mai importante sunt cele tehnice, programatice (legate de performanţe) şi de suportabilitate (legate de mediu). Dintre riscurile tehnice, se menţionează: schimbări tehnologice, schimbări ale calităţii materiei prime, limitări ale productivităţii, sau schimbări operaţionale. Alte riscuri interne, dar netehnice, sunt: întârzieri în aprovizionarea cu materii prime sau materiale, sau schimbări în echipa de operare. Unele riscuri sunt externe, dar nepredictibile, de pildă, inundaţii, cutremure, catastrofe, colaps financiar, sabotaj. Riscul trebuie privit din două perspective: pe termen scurt şi pe termen lung. În general, riscul este controlabil, în sensul că acţiunea umană îi poate modifica forma şi efectele, de pildă, elimina efectele negative şi întări efectele pozitive.

Siguranţă absolută, situaţia în care riscurile sunt complet eliminate, nu poate fi atinsă. Riscul poate fi doar redus la un nivel acceptabil. De aceea, toate riscurile trebuie tratate pe baza principiului ALARP (As Low As Reasonably Possible), adică urmărind obiectivul de a reduce nivelul de risc la valori cât mai mici posibil. Figura 5 ilustrează utilizarea nivelurilor de protecţie (echipamente şi/sau măsuri administrative) pentru a reduce riscul la un nivel acceptabil. În funcţie de acţiunea lor, nivelurile de protecţie pot fi clasificate în niveluri de prevenire şi niveluri de atenuare. Primele sunt folosite pentru a opri apariţia hazardului, iar celelalte pentru a reduce consecinţele apariţiei unui eveniment de hazard. În funcţie de locul în care acţionează, nivelurile de protecţie sunt împărţite în două categorii: în instalaţie şi în exteriorul acesteia. Metodele care asigură nivelurile de protecţie trebuie să fie independente, fiabile, proiectate special pentru riscul în discuţie şi să poată fi verificate.

Responsabilitatea funcţionării în deplină siguranţă a unei instalaţii tehnologice revine personalu-

lui de exploatare şi, în principal, inginerului de proces. Definirea riscurilor, identificarea metodelor şi mijloacelor de prevenire, cât şi stabilirea de proceduri şi strategii, se face încă din faza de proiectare. Identificarea, chiar din faza de concepţie, a riscului şi hazardului poate conduce la măsuri de implementare ce au în vedere reducerea sau eliminarea hazardului prin proiectare orientată către siguranţă [22]. Chiar dacă primele măsuri de siguranţă se iau din faza de proiectare, toate instalaţiile implementează şi strategii de gestionare a situaţiilor de risc în scopul eliminării riscului, sau, dacă acest lucru nu este posibil, pentru diminuarea efectelor negative ale unui eveniment neprevăzut.

4.1 Măsuri probabiliste pentru calculul riscului

Orice activitate industrială şi, în general, orice activitate de producţie, comportă elemente de incertitudine. Din punct de vedere matematic incertitudinea se modelează cu ajutorul variabilelor aleatoare sau, mai general, cu ajutorul proceselor stohastice.

Fie $(\Omega, \mathcal{K}, P)$ este un câmp de probabilitate, unde $(\Omega, \mathcal{K})$ este un câmp de evenimente ($\mathcal{K}$ este mulţimea părţilor mulţimii de selecţie Ω), iar P este o probabilitate pe $(\Omega, \mathcal{K})$. Se defineşte *risc* o variabilă aleatoare $X : \Omega \to \mathbb{R}$. Se noteaza cu $V[\Omega]$ mulţimea variabilelor aleatoare definite pe Ω. Fie M o submulţime a lui $V[\Omega]$. Se numeşte *măsură a riscului* o funcţie $f : M \to \mathbb{R}$. Prin $f(X)$ se noteaza măsura riscului X. Semnificaţia lui X poate fi legată de o serie de activităţi care au loc în cele mai diverse domenii şi care pot influenţa procesul industrial considerat. De exemplu, semnificaţia lui X poate fi: profitul activităţii industriale; valoarea pagubelor suferite la apariţia unei defecţiuni; depăşirea valorilor admisibile de către un parametru, în raport cu un un anumit prag (de pre-alarmă, de alarmă sau de plauzibilitate). Pentru a măsura riscul trebuie găsite modalităţi de a i se asocia un număr real. În cazul când semnificaţia lui X este efectul unei defecţiuni, atunci riscul apariţiei unui defect poate fi calculat prin una din următoarele formule:

$$f_1(X) = \sigma(X)^2 =: \mathrm{var}(X) = \mathcal{E}[(X-\mu)^2], \quad \mu := \mathcal{E}(X), \tag{1}$$

unde μ este *media*, σ^2 este *dispersia* (iar σ este *abaterea medie pătratică*), iar $\mathcal{E}$ este operatorul de medie;

$$f_2(X) = \mathcal{E}[(\tau - X)_+^p], \tag{2}$$

unde prin τ a fost notat un prag, numit *prag de avarie*, iar prin p, un factor de putere; indicele inferior $+$ denotă faptul că sunt luate în considerare numai valorile pozitive ale diferenţei $\tau - X$.

În cazul când semnificaţia lui X este depăşirea valorii maxime admise de către un paramentru în raport cu un un anumit prag, atunci riscul poate fi calculat prin următoarea formulă:

$$f_3(X) = \mathcal{E}[(X - \tau)_+^p]. \tag{3}$$

Media depăşirilor valorii parametrului peste valoarea acestui prag este considerată ca o măsură pentru riscul de funcţionare necorespunzătoare a unei instalaţii.

Dacă F_X este funcţia de repartiţie, iar f_X este densitatea de repartiţie de probabilitate a variabilei aleatoare X, atunci,

$$\mu(X) = \mathcal{E}(X) = \int_{-\infty}^{\infty} x \mathrm{d}F_X(x) = \int_{-\infty}^{\infty} x f_X(x) \mathrm{d}x, \quad \sigma(X)^2 = \int_{-\infty}^{\infty} (x-\mu)^2 \mathrm{d}F_X(x). \tag{4}$$

Dacă în locul integradului din formula dispersiei (4) se foloseşte modulul $|x-\mu|$, se obţine *abaterea medie absolută*, reprezentând *momentul centrat de ordinul 1*.

Alte măsuri ale riscului sunt prezentate, de exemplu, în [34] şi referinţele citate acolo.

Unele dintre formulele de mai sus pot fi implementate simplu. Dacă mulţimea de evenimente este finită, integralele devin sume. (Alternativ, se utilizează algoritmi numerici de integrare.) În

particular, media şi dispersia se calculează folosind formulele uzuale. O mare varietate de algoritmi pentru calcule statistice sunt implementate în bibliotecile comerciale IMSL (International Mathematics and Statistics Library), implementate în limbajele de programare C, Java, C#.NET şi Fortran. Este disponibilă, de asemea, şi o interfaţă la limbajul Python.

4.2 Tehnici pentru managementul riscului

Instrumente, practici şi protocoale pentru managementul riscului, permiţând identificarea şi evaluarea calitativă şi cantitativă a riscurilor în proiecte şi programe, din perspectiva responsabililor de proiecte, sunt prezentate pe larg în [32], într-o manieră predominant descriptivă. Sunt examinate într-o structură unitară peste 30 de tehnici de evaluare a riscului şi strategii de răspuns la situaţii de risc. Riscul este perceput ca fenomen viitor care se poate manifesta atât ca ameninţare, cât şi ca oportunitate. Oportunitatea variază în sens opus variaţiei riscului: un risc mic poate implica o mare oportunitate şi invers. Una dintre tehnicile des utilizate este *analiza SWOT* (Strengths, Weaknesses, Opportunities, and Threats), care permite identificarea riscului examinând punctele tari, cele slabe, oportunităţile şi ameninţările. O altă tehnică de mare reputaţie este *PERT* (Program Evaluation and Review Technique) în care se evaluează mediile şi dispersiile timpilor de execuţie a diferitelor activităţi dintr-un program sau proiect şi statisticile pentru cea mai scurtă durată de realizare (sau cel mai scurt traseu de parcurs, adică, *drumul critic*). Media duratei unei activităţi se calculează cu formula

$$(\text{optimist} + 4 \times (\text{cel mai probabil}) + \text{pesimist})/6,$$

iar dispersia cu

$$(\text{pesimist} - \text{optimist})/6,$$

unde "optimist" şi "pesimist" reprezintă estimaţiile corespunzătoare celui mai bun, respectiv, celui mai rău caz. Dispersia pentru drumul critic PERT se calculează cu formula

$$\sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2},$$

unde σ_i, $i = 1, \ldots, n$, sunt dispersiile asociate celor n componente ale drumului critic.

În general, un risc al unui eveniment cu mare probabilitate de apariţie şi mare impact trebuie tratat cu urgenţă, spre deosebire de alte riscuri cu probabilitate şi impact asociate mai mici. Riscul tehnologic şi fiabilitatea sunt adesea asociate cu ameninţări potenţiale ale vieţii sau mediului [14]. De pildă, întreruperea furnizării de energie electrică poate afecta serviciile de sănătate sau de intervenţii de urgenţă, traficul în oraşe mari, controlul traficului etc. Astfel de întreruperi sunt legate de decizii într-un sistem implicând risc, fiabilitate şi întreţinere. Adesea, aceste decizii includ mai multe obiective care trebuie tratate simultan, folosind modele multi-obiectiv sau multicriteriale.

Multe tehnici şi proceduri de modelare şi rezolvarea a problemelor de optimizare multicriteriale, inclusiv referitoare la analiza riscului şi identificarea hazardului, sunt prezentate în [14]. Când trebuie luată o decizie care implică risc, atunci este necesar să fie evaluate şi considerate consecinţele, probabilităţile de apariţie şi preferinţele decidentului. Întrucât consecinţele sunt adesea multidimensionale (de exemplu, cu dimensiunile: financiar, fiabilitate şi siguranţă, sau sănătate), este necesară găsirea unui compromis.

4.3 Analiza riscului şi identificarea hazardurilor

Analiza riscului poate fi definită ca o tehnică pentru identificarea, caracterizarea, cuantificarea şi evaluarea hazardului. Evaluarea riscului este un proces sistematic calitativ, cantitativ, sau semicantitativ care descrie natura, probabilitatea şi mărimea riscului asociat cu orice substanţă, situaţie, acţiune sau eveniment care include incertitudine.

Măsuri relevante pentru risc pot fi pierderile sau accidentele bazate pe date statistice, dar trebuie privite cu precauţie, deoarece multe dintre aceste statistici reprezintă medii, iar nu producerea unui accident specific cu pierderi potenţiale.

Proiectarea unei strategii de reducere sau eliminare a riscului începe cu analiza situaţiilor de hazard în proces. În faza de proiectare, aceasta conduce la revizuirea procesului tehnologic, a sistemului de control, a procedurilor de operare şi a celor de întreţinere. În etapa de operare, analiza riscurilor presupune evaluarea stării instalaţiei, identificarea riscurilor posibile şi intervenţia în scopul prevenirii situaţiilor nedorite. Utilizarea unei proceduri de evaluare a riscului şi hazardului, cum ar fi HAZOP, conduce la identificarea celor mai bune soluţii pentru sistemele cu siguranţă mărită [3]. Utilizarea mai multor metode de analiză a riscului cum sunt HAZOP, SIL, RCM şi RBI, asigură definirea unitară a riscului şi conduce în mod evident la creşterea siguranţei în funcţionare şi trebuie, pe cât posibil, să fie aplicată la fiecare proiect. Pe baza consecinţelor posibile, se poate efectua o clasificare a riscurilor în categorii de risc.

Efectuarea unui studiu HAZOP complex presupune o examinare detaliată şi riguroasă a procesului de către o echipă multidisciplinară incluzând ingineri de proces, de instrumentaţie, instalaţii electrice, instalaţii mecanice, specialişti în siguranţă, dar şi reprezentanţi ai beneficiarului. Sunt analizate diagramele cauză-efect şi riscurile aferente fiecărei funcţii şi operaţii din proces. Dacă în urma studiului se constată că sistemul de control al procesului nu este suficient pentru asigurarea condiţiilor cerute de beneficiar, se poate solicita un nivel SIS (Safety Instrumented System). Alocarea funcţiilor de siguranţă pe niveluri de protecţie este etapa care pune bazele întregului sistem de management pentru SIS, atribuind un nivel de siguranţă unei funcţii de siguranţă. Analiza SIL (Safety Integrity Level) este realizată având ca obiectiv alocarea SIL pentru fiecare funcţie SIF (Safety Instrumented Function). Standardele permit utilizarea mai multor metode: matricea riscurilor, graful riscurilor, analiza nivelurilor de protecţie sau arborele defecţiunilor. În cazul matricei riscurilor, care este cea mai utilizată metodă, se vor aloca niveluri SIL tuturor riscurilor, precum şi metode de protecţie adecvate încadrării în nivelul tolerabil de risc.

În [14] sunt descrise succint şi alte metode, cum ar fi analiza arborelui de avarii, analiza arborelui de evenimente, modelul avariei şi analiza efectului etc.

Pentru cuantificarea hazardurilor şi riscurilor, fiecărei categorii de risc îi corespund mai multe niveluri de consecinţe definite în funcţie de severitatea impactului pe care îl are apariţia unui risc din categoria respectivă. De asemenea, fiecărui tip de consecinţă i se asociază un nivel de probabilitate de apariţie în intervalul de timp de referinţă. (Uzual, acest interval este durata de viaţă a instalaţiei.)

4.4 Analiza alarmelor

Sistemele de alarmare din cadrul unei instalaţii constituie o parte esenţială a interfeţei operator de monitorizare şi control, întrucât gestionarea corectă a alarmelor într-o situaţie cu risc ridicat presupune acţiuni de reducere a nivelului de risc. Cu toate că majoritatea alarmelor permit unui operator să detecteze o funcţionare anormală şi să identifice cauza acesteia, cele mai multe aplicaţii de monitorizare vor afişa o cantitate foarte mare de alarme într-o situaţie de risc ridicat, ceea ce va duce la o capacitate scăzută din partea operatorilor de a putea identifica în mod corect cauza acestora. Astfel, alarmele critice se pierd printre cele care nu sunt neapărat necesare. Pentru evaluarea eficientă a stării de alarmă dintr-o instalaţie, precum şi a riscului asociat, este necesară analiza alarmelor la nivel de sistem, prin monitorizarea şi gestionarea continuă a acestora folosind algoritmi dedicaţi. În practică, sunt folosiţi mai mulţi indicatori de performanţă (KPI — key performance indicator) pentru evaluarea nivelului de performanţă al unui sistem de alarmare, precum: media ratei de alarmare, rata maximă de alarmare, sau procentul de timp în care ratele de alarmare sunt în afara nivelului acceptat. Rata medie de alarmare se calculează ca fiind numărul total de alarme raportat la numărul

total de perioade de timp. Se recomandă ca acest raport într-o operare normală să fie sub 1 la 10 minute. Rata maximă de alarmare reprezintă numărul de alarme care îi apar unui operator în timpul oricărui interval de 10 minute. Dacă acest raport are o valoare de 10 alarme în 10 minute înseamnă că sistemul nu poate fi gestionat corect de către un operator pe perioade mari de timp. Metoda "cele mai rele 10 alarme" a fost folosită pe scară largă în aplicaţii industriale în Japonia pentru a reduce numărul de alarme care nu sunt necesare. Această metodă este utilizată pentru a colecta date din istoricul de alarme în timpul operării, pentru a crea o listă a celor mai frecvent generate alarme. Deşi această metodă poate reduce efectiv numărul de alarme pe care le vizualizează un operator, este mai puţin eficientă când vine vorba de reducerea numărului de alarme pe măsură ce dispar unele dintre cele zece cele mai rele alarme.

Nishiguchi şi Takai [28] au propus o metodă de evaluare bazată pe istoricul datelor, care se referă nu numai la analiza alarmelor, ci şi a evenimentelor apărute în funcţionarea instalaţiei. Datele de operare sunt convertite în evenimente secvenţiale, $s_i(k)$. Când apare evenimentul i între momentele $(k-1)\Delta t$ şi $k\Delta t$, atunci $s_i(k) = 1$, dar altfel $s_i(k) = 0$. Funcţia de intercorelare, $c_{ij}(m)$, dintre $s_i(k)$ şi $s_j(k)$, pentru intervalul de timp m, se calculează cu ecuaţia (5). Aici, K este perioada de timp maximă pentru durata intervalului şi T este durata totală de timp pe parcursul căreia au fost preluate datele.

$$c_{ij}(m) = \begin{cases} \sum_{k=1}^{\frac{T}{\Delta t}-m} s_i(k)s_j(k+m), & 0 \le m \le \frac{K}{\Delta t} \\ c_{ij}(-m), & -\frac{K}{\Delta t} \le m < 0 \end{cases} . \tag{5}$$

Când două evenimente, i şi j, sunt independente unele de altele, probabilitatea ca ele să apară simultan de mai mult de c_{ij}^* ori este dată de funcţia de intercorelare pe intervalul de timp m din ecuaţia (6) .

$$P\left(c_{ij}(m) \ge c_{ij}^* \,\middle|\, -\frac{K}{\Delta t} \le m \le \frac{K}{\Delta t}\right) \approx 1 - \left(\sum_{l=0}^{c_{ij}^*-1} \frac{\mathrm{e}^{-\lambda}\lambda^l}{l!}\right)^{2K+1} . \tag{6}$$

5 Algoritmi de detecţie a avariilor, diagnoză şi acomodare

Raportul sintetizează şi o problematică de mare interes şi actualitate, atât pe plan teoretic, cât şi pe plan practic, şi anume, toleranţa la defecte a sistemelor automate. Această direcţie de cercetare s-a dezvoltat iniţial cu precădere în domeniul aeronautic, având aplicaţii la conducerea automată a aeronavelor. Totuşi, cercetările s-au extins şi în alte domenii, de pildă, în industria chimică şi petrochimică, energetica nucleară etc. Domeniul abordat a înregistrat o dezvoltare deosebită în ultimii ani, datorită importanţei asigurării securităţii în funcţionarea acestor sisteme.

Conducerea optimală robustă şi ajustarea modelului admisibil oferă o perspectivă tolerantă la defecte asupra adaptării modelului pentru a coincide cu, sau a aproxima, un model de referinţă şi oferă o abordare admisibilă pentru recuperarea după defecte. În context, model de referinţă poate fi modelul nominal, în absenţa defectelor. Este esenţială asigurarea stabilităţii şi robusteţei, dar şi a eficienţei computaţionale, pentru implementarea în timp real. Se discută o tehnică modificată de pseudo-inversă şi se compară rezultatele cu cele obţinute folosind metoda cu urmărire a modelului optimal robust, bazată pe un indice de performanţă cu factor exponenţial, asigurând un grad de stabilitate impus. Tehnica propusă garantează stabilitatea sistemului post-avarie. Se consideră atât acomodarea progresivă la defecte, cât şi aproximarea admisibilă şi optimală.

5.1 Detecţia avariilor

Orice sistem fizic construit de om şi, cu atât mai mult, un sistem complex, poate să se defecteze. În anumite cazuri, defectarea poate avea consecinţe dezastruoase. De aceea, detecţia, izolarea şi identificarea/estimarea defectelor şi *conducerea tolerantă la defecte/avarii* (Fault Tolerant Control—FTC) sunt funcţionalităţi necesare pentru asigurarea operării adecvate şi în condiţii de siguranţă a sistemelor automate complexe. Folosind abordarea FTC un sistem automat poate poseda proprietăţile dorite atât în funcţionarea normală, cât şi în prezenţa avariilor. Proprietăţi dorite des folosite sunt *concordanţa modelului* (model matching) sau *urmărirea modelului* (model following). Concordanţa modelului presupune că matricea de transfer a sistemului în buclă închisă coincide cu matricea de transfer în buclă închisă a unui *model de referinţă*. Ideal, urmărirea modelului înseamnă identitatea traiectoriilor de stare ale sistemului şi modelului. Nici una dintre aceste proprietăţi nu poate fi obţinută în general şi sunt necesare aproximaţii.

Sistemele tolerante la defecte şi sistemele optimale reconfigurabile au constituit subiectul unor cărţi recent publicate şi al multor articole apărute în reviste ştiinţifice prestigioase [1],[11]–[13],[20, 21, 23, 26, 27],[39]–[42], [47, 48].

Conducerea avansată a unui proces implică acumularea continuă de date de măsură a variabilelor procesului, cât şi transmiterea, memorarea şi prelucrarea lor, pentru supraveghere (monitorizare), modelare, predicţie, administrarea mai bună a resurselor etc. Sunt parcurse diverse etape: achiziţia şi memorarea datelor, prelucrarea primară, extragerea informaţiei, agregarea datelor, modelarea etc. Trebuie luate în considerare diverse aspecte, incluzând eterogeneitatea, timpul de prelucrare, securitatea şi caracterul privat al datelor, interacţiunea cu operatorul uman. Extragerea informaţiei şi cunoaşterii din mulţimi de date foloseşte actualmente tehnici de căutare a formelor sau tendinţelor (data mining) sau metode de învăţare statistice. O metodă care capătă o mare reputaţie, în special în contextul prelucrării volumelor mari de date (“Big data”) este *învăţarea în adâncime* (“deep (machine) learning”), numită şi *învăţare ierarhică*. Se utilizează niveluri de prelucrare multiple, cu structuri complexe, compuse din reprezentări cu transformări neliniare (implementate, de pildă, în reţele neurale). Această abordare permite extragerea şi învăţarea eficientă şi nesupravegheată sau semi-supravegheată a trăsăturilor.

Observaţiile (statistice) *deviante*, clar diferite ca valoare de altele dintr-un eşantion, numite în continuare şi*anomalii* (outliers), semnifică uneori riscuri sau oportunităţi. Sunt necesare mijloace pentru detecţia lor. O abordare posibilă se bazează pe testarea ipotezelor statistice. Observaţiile deviante pot fi cauzate de abateri trecătoare în cursul achiziţiei datelor, datorate funcţionării necorespunzătoare a aparatelor de măsură, zgomotelor (inclusiv pe canalele de transmitere), sau schimbărilor abrupte ale naturii sau comportării procesului. Aceste observaţii trebuie eliminate, de pildă, dacă se doreşte modelarea funcţionării normale a unui proces, dar trebuie analizate când ar putea semnala o comportare anormală, posibil conducând la regimuri de funcţionare critice, periculoase sau inacceptabile. Modelarea observaţiilor deviante şi abstractizarea problemei detecţiei acestora au trei componente importante: nivelul informaţiei disponibile despre comportarea normală şi deviantă, tipul deviaţiilor şi criteriul pentru identificarea acestora. Abordările existente pentru detecţia anomaliilor pot fi clasificate în patru grupe: supervizate, semisupervizate, nesupervizate şi complet universale. Abordările supervizate sunt aplicabile când sunt disponibile modele atât pentru observaţiile normale, cât şi pentru cele deviante, ceea ce este posibil pentru date statice sau modele lent variabile în timp. Această clasă include abordări bazate pe clasificare, reţele neurale, Bayes şi support vector machines (SVM). Abordările semisupervizate folosesc doar un model, fie al datelor normale (în majoritatea cazurilor), fie al celor deviante. Abordările nesupervizate nu utilizează nici o ipoteză despre modelele datelor; exemple sunt abordările discriminative, abordări parametrice şi prelucrarea analitică on-line. Abordările complet universale construiesc reguli de decizie cu singura

ipoteză că distribuţiile normală şi deviantă sunt diferite.

O prezentare recentă a problematicii de mai sus pentru seturi mari de date se găseşte în [38]. Instrumentul principal folosit este testarea ipotezelor statistice.

5.2 Algoritmi de reconfigurare şi acomodare post-avarie

Mai des folosite au fost două politici pentru FTC, reconfigurarea sistemului şi acomodarea la defecte, depinzând de excluderea sau, respectiv, includerea subsistemelor afectate de defecte din/în proiectarea noii legi de comandă, aplicată post-avarie. Obiectivul este fie minimizarea unui criteriu de optim integral (urmărirea modelului), fie aproximarea cât mai bună a modelului ideal (ideal model matching). Una dintre soluţii este schema de "acomodare" progresivă pentru problema de conducere liniar-pătratică post-avarie şi "suprapunerea" simplificată şi extinsă peste modelul optimal robust pentru problema de urmărire a modelului optimal post-avarie [7]–[13].

Se consideră un sistem deterministic liniar invariant în timp (LTI) a cărui comportare nominală, în absenţa perturbaţiilor şi avariilor, este descrisă de modelul de stare [13]

$$M_n: \quad \dot{x}(t) = A_n x(t) + B_n u_n^*(t), \qquad x(0) = x_0, \tag{7}$$

unde $A_n \in \mathbb{R}^{n\times n}$ and $B_n \in \mathbb{R}^{n\times m}$ sunt matricele nominale de stare şi, respectiv, de comandă, $x(t)$ şi $u_n^*(t)$ sunt vectorii de stare şi de comandă la momentul t, iar x_0 este starea iniţială. Pentru simplitate, se presupune că vectorul de ieşire a sistemului, $y(t)$, coincide cu $x(t)$. Se consideră, de asemenea, că în funcţionarea normală, se foloseşte legea de comandă cu reacţie de la stare

$$u_n^*(t) = -K_n^* x(t), \tag{8}$$

astfel încât sistemul în buclă închisă, descris de formula

$$M_n^*: \quad \dot{x}(t) = \widehat{A}_n^* x(t), \qquad \widehat{A}_n^* = A_n - B_n K_n^*, \tag{9}$$

este stabil şi are comporatarea dinamică dorită. Modelul M_n^* din (9) este considerat *modelul de referinţă*, a cărui dinamică ar trebui reprodusă cât mai fidel posibil în orice alte regimuri de funcţionare, diferite de funcţionarea ideală. În plus, se impune asigurarea unui *grad de stabilitate* dorit, $\alpha_n^* > 0$, adică $\Re(\lambda_i) < -\alpha_n^*$, $i = 1, \ldots, n$, unde λ_i sunt *polii* sistemului în buclă închisă (valorile proprii ale matricei de stare a acestui sistem).

Apariţia unei modificări parametrice a sistemului, de pildă, ca urmare a unei defecţiuni, sau a schimbării regimului de funcţionare, face ca modelul nominal (7) să nu mai fie adecvat. Fie

$$M: \quad \dot{x}(t) = Ax(t) + Bu(t), \tag{10}$$

modelul identificat sau estimat de un algoritm de detecţie, izolare şi identificare/estimare după producerea schimbării înregistrate în proces, şi fie legea de comandă

$$u(t) = -Kx(t). \tag{11}$$

Realizarea concordanţei exacte cu (9) a sistemului în buclă închisă definit de (10) şi (11) ar presupune îndeplinirea condiţiei:

$$\widehat{A} := A - BK = \widehat{A}_n^*, \tag{12}$$

ceea ce atrage cerinţa ca subspaţiul liniar generat de coloanele matricei $\widehat{A} - \widehat{A}_n^*$ să fie inclus în subspaţiul liniar generat de coloanele matricei B. Evident, această cerinţă nu poate fi satisfăcută

în general. Ca atare, problema concordanţei (exacte) nu are soluţie în general. Se poate obţine o soluţie aproximativă, folosind pseudo-inversa Moore-Penrose,

$$K^{\text{PI}} = B^{\dagger}(A - \widehat{A}_n^*). \tag{13}$$

În practică, nu se calculează pseudo-inversa $B^{\dagger}$, ci se rezolvă, în sensul celor mai mici pătrate, sistemul de ecuaţii algebrice liniare $BK = A - \widehat{A}_n^*$, folosind, de pildă, descompunerea după valorile singulare ale matricei B. Dar rezolvarea aproximativă a problemei concordanţei modelului poate însă conduce la sisteme în buclă închisă instabile.

De fapt, nu este necesară impunerea condiţiei de concordanţă a modelelor. Comportarea dinamică a sistemului definit de (7) şi (8) este complet descrisă de spectrul matricei $\widehat{A}_n^*$, $\Lambda(\widehat{A}_n^*)$. Dacă sistemul (10) este controlabil, se poate obţine o matrice K astfel încât $\widehat{A}$ din (12) să aibă spectrul $\Lambda(\widehat{A}_n^*)$, folosind algoritmi de *alocare a polilor*. Un studiu recent al unor algoritmi de alocare *robustă* a polilor (asigurând, de pildă, sensibilitate redusă la perturbaţii ale matricelor sistemului) este [30]. O deficienţă a algoritmilor de alocare a polilor este că matricea de reacţie K poate avea normă mare, ceea ce poate determina valori posibil inacceptabil de mari ale elementelor vectorului de comandă $u(t)$. Reducerea normei matricei K se poate efectua folosind proceduri de optimizare, fără a reduce semnificativ robusteţea. Alternativ, se poate renunţa la plasarea exactă a polilor în locaţii predefinite şi realiza plasarea lor în regiuni specificate ale planului complex $\mathbb{C}$, de pildă, la stânga abscisei $-\alpha_n^*$, asigurând astfel gradul de stabilitate dorit. Acest obiectiv poate fi atins rezolvând o problemă de optimizare liniar-pătratică cu ponderare exponenţială în criteriul de optim.

O soluţie a problemei menţionate mai sus în cazul $m = 1$ este prezentată în [13]. Se utilizează metoda modificată a pseudo-inversei. Componentele vectorului de reacţie corespunzător, $K^{\text{MPI}} = \left(k_j^{\text{MPI}}, j = 1, \ldots, n\right)$, sunt definite astfel:

$$k_j^{\text{MPI}} = \begin{cases} k_j^{\text{PI}}, & \text{adică } |k_j^{\text{PI}}| \le \widetilde{\delta}^{\text{PI}}, \\ \text{sgn}(k_j^{\text{PI}})\widetilde{\delta}^{\text{PI}}, & \text{în caz contrar,} \end{cases} \tag{14}$$

unde $\widetilde{\delta}^{\text{PI}}$ reprezintă marginea de stabilitate robustă truncheată, calculată cu formula $\widetilde{\delta}^{\text{PI}} = \delta^{\text{PI}} - \epsilon^{\text{PI}}$, pentru o valoare mică ϵ^{PI}, iar sgn$(\cdot)$ reprezintă funcţia semn a unui argument real. Calculul lui δ^{PI} este discutat în [13].

Formulele (14) sunt motivate de faptul că, dacă sistemul în buclă închisă folosind reacţia de stare cu vectorul K^{PI} din (13) este instabil, utilizarea vectorului K^{MPI} din (14) îl face stabil, prin definiţia marginii de stabilitate; într-adevăr, sistemul închis devine

$$\dot{x}(t) = \widehat{A}^{\text{MPI}} x(t), \quad \widehat{A}^{\text{MPI}} := A - BK^{\text{MPI}} = A - \delta^{\text{PI}} BK^{\text{PI}}, \tag{15}$$

şi se poate arăta că $|k_j^{\text{MPI}}| \le \widetilde{\delta}^{\text{PI}}$ garantează stabilitatea.

Soluţia propusă în [13] pentru acomodarea post-avarie foloseşte stabilizarea determinată prin rezolvarea unei ecuaţii algebrice Riccati continue generalizate (corespunzătoare unui criteriu de optimizare integral matrice de ponderare determinate adecvat şi asigurând o margine de stabilitate dorită) şi o corecţie cu metoda modificată a pseudo-inversei. Pentru rezolvarea ecuaţiei Riccati se poate folosi o metodă iterativă de tip Newton [36, 37], care are avantajul că are o convergenţă pătratică în vecinătatea soluţiei, iar iteratele sunt stabilizatoare (începând de la a doua). Prima proprietate permite obţinerea soluţiei în două-trei iteraţii, dacă procedura Newton este iniţializată adecvat (de exemplu, folosind soluţia ecuaţiei Riccati pentru regimul de funcţionare anterior); în acest caz, nu este necesară adoptarea variantei cu căutare liniară a metodei Newton. Ultima proprietate permite folosirea unei soluţii suboptimale, obţinute înaintea terminării procesului iterativ. Ambele proprietăţi fac ca timpul de calcul să fie relativ redus, astfel încât să fie acceptabil pentru o conducere în timp real.

6 Concluzii

Prezentarea făcută în acest raport dovedeşte că obiectivele propuse pentru această etapă a proiectului, Etapa II, au fost atinse în întregime. Toate cele patru activităţi prevăzute au fost efectuate. Investigaţiile întreprinse se vor dovedi foarte utile pentru realizarea etapelor următoare.

Implementarea aplicaţiilor de conducere automată avansată a proceselor industriale necesită uzual resurse sporite de stocare şi execuţie. Raportul a prezentat o aplicaţie bazată pe cloud care îi permite unui utilizator să aibă acces la diferite strategii de conducere folosind o interfaţă web, să beneficieze de regulatoare virtualizate, care să execute acei algoritmi şi să trimită comenzi dispozitivelor din proces. Raportul a definit cum modulele implementând aceste noi servicii pot fi interconectate utilizând interfeţe RESTful. Abordarea inovativă prezentată presupune virtualizarea unor "baze de date" de regulatoare de proces şi îi conferă inginerului automatist dintr-o întreprindere industrială accesul la strategii avansate de modelare, optimizare şi conducere, implementate ca funcţii bloc IEC 61499.

Au fost investigate, de asemenea, aspecte referitoare la managementul situaţiilor de risc, cât şi pentru detecţia avariilor, diagnoză şi acomodare.

Etapa II nu a presupus diseminarea explicită a unor rezultate. Totuşi, au fost publicate cinci lucrări elaborate de unii membri ai echipei proiectului (una dintre ele cu alţi doi colaboratori, care nu fac parte din echipă), iar o altă lucrare va fi prezentată la o conferinţă care va avea loc la Budapesta în luna decembrie 2015 şi va fi publicată de către WSEAS (World Scientific and Engineering Academy and Society). Lucrările publicate deja au fost prezentate la conferinţe internaţionale co-sponsorizate de IEEE (Institute of Electrical and Electronics Engineers) şi sunt incluse în reputata colecţie IEEE Xplore Digital Library. Volumele de lucrări sunt în curs de indexare de către ISI Thomson (ISI Proceedings). Toate cele şase lucrări menţionează într-o secţiune de mulţumiri suportul financiar parţial din proiectul CALCULOS.

Bibliografie

[1] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, editors. *Diagnosis and Fault Tolerant Control*. Springer-Verlag, 2006.

[2] T. Bray. The JavaScript object notation (JSON) data interchange format, 2014. [Online]. Available: `http://tools.ietf.org/html/rfc7159.html`. [Accessed: 22-Nov-2015].

[3] Center for Chemical Process Safety, American Institute of Chemical Engineers. Introduction to inherently safer design, 2009.

[4] O. Chenaru, G. Stamatescu, I. Stamatescu, and D. Popescu. Towards cloud integration for industrial wireless sensor network systems. In *Proceedings of the 9th International Symposium on Advanced Topics in Electrical Engineering, 7-9 May, 2015, Bucharest, Romania*, pages 917–922, 2015.

[5] O. Chenaru, A. Stanciu, D. Popescu, G. Florea, V. Sima, and R. Dobrescu. Modeling complex industrial systems using cloud services. In *Proceedings of The 20th International Conference on Control Systems and Computer Science (CSCS20), May 27-29, 2015, Bucharest, Romania*, pages 565–571, 2015.

[6] O. Chenaru, A. Stanciu, D. Popescu, G. Florea, V. Sima, and R. Dobrescu. Open cloud solution for integrating advanced process control in plant operation. In *Proceedings of the 23rd Mediterranean Conference on Control and Automation (MED 2015), Torremolinos, Spain, June 16th-19th, 2015*, pages 973–978, 2015.

[7] B. Ciubotaru and M. Staroswiecki. Fault tolerant control of the B747 short-period mode using progressive accommodation. In *Proceedings of the 2006 IEEE International Conference on Control Applications, Munich, Germany, October 4-6*, pages 3288–3294, 2006.

[8] B. Ciubotaru, M. Staroswiecki, and C. Christophe. Fault tolerant control of the boeing 747 short-period mode using the admissible model matching technique. In *Proceedings of the 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Tsinghua University, P.R. China, August 29 - September 1, 2006*, pages 819–824, 2006.

[9] B. D. Ciubotaru. *Perspective on Fault Tolerance in Flight Control*. Phd thesis, Department of Automatic Control and Computer Science, Polytechnic University of Bucharest, Bucharest, Romania, Nov. 2009.

[10] B. D. Ciubotaru and M. Staroswiecki. Anytime algorithm for parametric faults accommodation under handling quality constraints. In *Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Barcelona, Spain, June 30 - July 3, 2009*, pages 887–892, 2009.

[11] B. D. Ciubotaru and M. Staroswiecki. Comparative study of matrix Riccati equation solvers for parametric faults accommodation. In *Proceedings of the 10th European Control Conference, 23-26 August 2009, Budapest, Hungary*, pages 1371–1376, 2009.

[12] B. D. Ciubotaru and M. Staroswiecki. Extension of modified pseudo-inverse method with generalized linear quadratic stabilization. In *Proceedings of the 2010 American Control Conference, Marriott Waterfront, Baltimore, MD, USA, June 30-July 02, 2010*, pages 6222–6224, 2010.

[13] B. D. Ciubotaru, M. Staroswiecki, and N. D. Christov. Modified pseudo-inverse method with generalized linear quadratic regulator for fault tolerant model matching with prescribed stability degree. In *Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), Orlando, FL, USA, December 12-15, 2011*, pages 1583–1588, 2011.

[14] A. T. de Almeida, C. A. V. Cavalcante, M. H. Alencar, R. J. P. Ferreira, A. T. de Almeida-Filho, and T. V. Garcez. *Multicriteria and Multiobjective Models for Risk, Reliability and Maintenance Decision Analysis*, volume 231 of *International Series in Operations Research & Management Science,* Camille C. Price and Joe Zhu (Series editors). Springer, Heidelberg New York Dordrecht London, 2015.

[15] FBDK client server — Google groups. [Online]. Available: `http://groups.google.com/forum/#!msg/fbdk/storw0Op3vo/J5S8q6nMAQUJ`. [Accessed: 22-Nov-2015].

[16] G. Florea, O. Chenaru, D. Popescu, and R. Dobrescu. A fractal model for power smart grids. In *Proceedings of The 20th International Conference on Control Systems and Computer Science (CSCS20), May 27-29, 2015, Bucharest, Romania*, pages 572–577, 2015.

[17] G. Florea, O. Chenaru, D. Popescu, and R. Dobrescu. Evolution from power grid to smart grid: Design challenges. In S. Caraman, M. Barbu, and R. Solea, editors, *Proceedings of the 19th International Conference on System Theory, Control and Computing, October 14-16, 2015, Cheile Gradistei - Fundata Resort, Romania*, pages 912–916, 2015.

[18] O. Givehchi, H. Trsek, and J. Jasperneite. Cloud computing for industrial automation systems — A comprehensive overview. In *Proceedings of the 18th IEEE International Conference on Emerging Technologies and Factory Automation, 10-13 Sep 2013, Cagliari, Italy*, pages 1–4, 2013.

[19] T. Goldschmidt, A. Jansen, H. Koziolek, J. Doppelhamer, and H. P. Breivold. Scalability and robustness of time-series databases for cloud-native monitoring of industrial processes. In *Proceedings of the IEEE 7th International Conference on Cloud Computing, June 27 - July 2, 2014, Alaska, USA*, pages 602–609, 2014.

[20] I. Hwang, S. Kim, Y. Kim, and C. E. Seah. A survey of fault detection, isolation and reconfiguration methods. *IEEE Trans. on Control Systems Techn.*, 18(3):636–653, 2010.

[21] R. Isermann. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5):709–719, 1997.

[22] A. Jaya. Safety in overpressure relieving systems (Engineering design guideline). Technical report, KLM Technology Group, Practical Engineering Guidelines for Processing Plant Solutions, 81200 Johor Bahru, Malaysia, 2011. Rev: 01. Available: `http://kolmetz.com/pdf/EDG/ENGINEERING_DESIGN_GUIDELINE_safety_over_pressure_Rev_web.pdf`.

[23] J. Jiang. Fault-tolerant control systems -- An introductory overview. *Acta Automatica Sinica*, 31(1):161–174, 2005.

[24] JSON — Wikipedia, the free encyclopedia. [Online]. Available: `https://en.wikipedia.org/wiki/JSON`. [Accessed: 22-Nov-2015].

[25] Kubernetes by Google. [Online]. Available: `http://kubernetes.io`. [Accessed: 22-Nov-2015].

[26] J. Lunze and J. H. Richter. Reconfigurable fault-tolerant control: A tutorial introduction. *European J. of Control*, 14(5):359–386, 2008.

[27] J. F. Magni, S. Bennani, and J. Terlouw, editors. *Robust Flight Control: A Design Challenge*. Springer-Verlag, 1997.

[28] J. Nishiguchi and T. Takai. IPL2 and 3 performance improvement method for process safety using event correlation analysis. *Computers & Chemical Engineering*, 34(12):2007–2013, 2007.

[29] L. A. Ocheană, O. I. Rohat, D. A. Popescu, and G. G. Florea. Library of reusable algorithms for Internet-based diagnose and control system. In *Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing, 23-25 May 2012, Bucharest, Romania,* Part 1, pages 1407–1412, 2012. Available: `http://www.ifac-papersonline.net/Detailed/53961.html`.

[30] A. Pandey, R. Schmid, T. Nguyen, Y. Yang, V. Sima, and A. L. Tits. Performance survey of robust pole placement methods. In *Proceedings of the 2014 IEEE 53rd Annual Conference on Decision and Control (CDC 2014), Los Angeles, California, USA, December 15–17 2014*, pages 3186–3191. IEEE, 2014.

[31] Pods. [Online]. Available: `http://kubernetes.io/v1.1/docs/user-guide/pods.html`. [Accessed: 22-Nov-2015].

[32] C. L. Pritchard. *Risk Management. Concepts and Guidance*. CRC Press, Boca Raton, FL 33487-2742, fifth edition, 2015.

[33] O. Rohat and D. Popescu. Remote web-based execution of IEC 61499 function blocks. In *Proceedings of the 6th International Conference on Electronics, Computers and Artificial Intelligence, Oct 23-25, 2014, Bucharest, Romania*, pages 33–38, 2014.

[34] M. Rădulescu, S. Rădulescu, and C. Z. Rădulescu. *Modele Matematice pentru Optimizarea Investiţiilor Financiare*. Editura Academiei Române, Bucureşti, 2006.

[35] Serialization — Wikipedia, the free encyclopedia. [Online]. Available: `https://en.wikipedia.org/wiki/Serialization`. [Accessed: 22-Nov-2015].

[36] V. Sima. Efficient computations for solving algebraic Riccati equations by Newton's method. In M. H. Matcovschi, L. Ferariu, and F. Leon, editors, *Proceedings of the 2014 18th Joint International Conference on System Theory, Control and Computing (ICSTCC 2014), October 17-19, 2014, Sinaia, Romania*, pages 609–614, 2014. ISSN 978-1-4799-4602-0.

[37] V. Sima and P. Benner. Numerical investigation of Newton's method for solving continuous-time algebraic Riccati equations. In J.-L. Ferrier, O. Gusikhin, K. Madani, and J. Sasiadek, editors, *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2014), 1-3 September, 2014, Vienna, Austria*, volume 1, pages 404–409. SciTePress — Science and Technology Publications, Portugal, 2013. ISSN 978-989-758-039-0.

[38] A. Tajer, V. V. Veeravalli, and H. V. Poor. Outlying sequences detention in large data sets. *IEEE Signal Proc. Mag.*, 31(5):44–56, Sept. 2014.

[39] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri. A review of process fault detection and diagnosis. Part II. Qualitative models and search strategies. *Computers and Chemical Eng.*, 27(3):313–326, 2003.

[40] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin. A review of process fault detection and diagnosis. Part III. Process history based methods. *Computers and Chemical Eng.*, 27(3):327–346, 2003.

[41] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis. Part I. Quantitative model-based methods. *Computers and Chemical Eng.*, 27(3):293–311, 2003.

[42] M. Verhaegen, S. Kanev, R. Hallouzi, C. Jones, J. Maciejowski, and H. Smaili. Fault tolerant flight control – A survey. In C. Edwards, T. Lombaerts, and H. Smaili, editors, *Fault-Tolerant Flight Control – A Benchmark Challenge*, chapter 2, pages 47–89. Springer-Verlag, 2010.

[43] V. Vyatkin. IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review. *IEEE Trans. Ind. Inform.*, 7:768–781, 2011.

[44] YAML — Wikipedia, the free encyclopedia. [Online]. Available: `https://en.wikipedia.org/wiki/YAML`. [Accessed: 22-Nov-2015].

[45] YAML ain't markup language (YAML™) Version 1.2. [Online]. Available: `http://www.yaml.org/spec/1.2/spec.html`. [Accessed: 22-Nov-2015].

[46] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[47] Y. Zhang and J. Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2):229–252, 2008.

[48] A. Zolghadri. The challenge of advanced model-based FDIR techniques for aerospace systems: The 2011 situation. In *Proc. 4th European Conference for Aerospace Sciences*, 2011.