**Program PARTENERIATE - Proiecte colaborative de cercetare aplicativă - Competiţie 2013**

# RST - Raport ştiinţific şi tehnic extins

# 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 I – Cerinţele utilizatorului, analiza acceptabilităţii şi platforma colaborativă

**Decembrie 2014**

# Cuprins

# 1. Introducere

## 1.1. Obiectivul general

**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, defectoscopie, diagnoză, prevenire a avariilor şi analiza defecţiunilor.

**Obiectivele specifice** ale proiectului sunt:

- Proiectarea unei platforme care să folosească o interfaţă prietenoasă de programare adresată diferitelor tipuri de utilizatori: ingineri software, ingineri specializaţi în identificarea parametrică şi modelare matematică, ingineri specializaţi în controlul proceselor (automatişti) care lucrează cu algoritmi complecşi, inclusiv algoritmi genetici şi reţele neuronale;
- Crearea unor maşini virtuale care să poată găzdui module/aplicaţii, algoritmi de simulare şi optimizare;
- Reducerea costurilor de mentenanţă pentru instalaţiile industriale;
- Îmbunătăţirea relaţiei între mediul academic şi cel industrial;
- Îmbunătaţirea proceselor şi instalaţiilor industriale româneşti prin metode şi servicii accesibile.

Proiectului îşi propune să abordeze integrat aspectele de optimizare şi securitate/siguranţă a funcţionării proceselor, aspecte tratate în mod separat mai înainte. De asemenea, pentru a veni în sprijinul inginerilor automatişti din industrie, se doreşte utilizarea unor funcţii bloc standardizate pentru încapsularea de algoritmi şi strategii de control. Schemele actuale de control de proces sunt implementate prin echipamente hardware şi software dedicate, care adesea fac foarte dificilă sau chiar imposibilă implementarea îmbunătăţirilor sau adaptărilor. Mai mult, utilizarea puterii de calcul cloud pentru dezvoltarea, testarea şi validarea unor noi algoritmi şi strategii de control reprezintă o provocare majoră în domeniul proiectării şi analizei sistemelor de control, şi bătătoreşte calea pentru îmbunătăţirea continuă a calităţii, flexibilităţii, fiabilităţii şi eficienţei proiectelor şi implementărilor.

**CALCULOS propune o platformă bazată pe cloud pentru servicii de proiectare şi validare ce vor asigura suport pentru o bază de date deschisă şi acces la soluţii algoritmice standardizate.** Baza de date va consta în algoritmi, secvenţe şi strategii de control optimizate pentru instalaţii şi procese specifice, permiţând o creştere substanţială a eficienţei şi fiabilităţii, reducând riscul de apariţie a funcţionării defectuoase şi facilitând evaluarea experimentală a diverselor scenarii. **Totuşi, identificarea unor soluţii potrivite reprezintă o altă provocare majoră** [1]. În acest proiect va fi dezvoltată o soluţie, bazată pe căutări semantice şi potrivirea algoritmilor cu proprietăţile cerute, conducând la modificarea sistemului de control. Verificarea la nivelul sistemului va fi făcută în cloud de către modulul simulare-în-buclă, accesat de către dezvoltatori folosind interfaţa web pusă la dispoziţie. Proiectul tratează unele din cele mai importante provocări în domeniul sistemelor distribuite de control la scară largă şi se foloseşte de cele mai avansate tehnici

din domeniile **ingineriei procesării în cloud, serviciilor bazate pe internet, funcţiilor bloc refolosibile standardizate**, **optimizării sistemelor complexe**, **controlului tolerant la erori şi analiza de pericole,** pentru a crea algoritmi noi, eficienţi şi fiabili, care vor fi amplasaţi într-o bază de date deschisă, răspunzând cerinţelor actuale de control din industrie [2, 3]. Ţintirea către ambele obiective, optimizarea şi siguranţa/securitatea, reprezintă o mare provocare, greu de atins cu instrumente clasice. Este necesară elaborarea unui cadru capabil să ofere nu numai instrumente şi algoritmi, dar şi procedurile şi capabilităţile de calcul pentru a rula în timp real strategii complexe şi algoritmi, inclusiv simulări, evaluări de risc şi optimizări. Proiectul va dezvolta o arhitectură capabilă să satisfacă aceste obiective.

### 1.2. Obiectivul etapei de execuţie

Conform planului de realizare a proiectului, în cadrul acestei prime etape de execuţie a proiectului, *Etapa I – Cerinţele utilizatorului, analiza acceptabilităţii şi platforma colaborativă*, au fost efectuate de către parteneri următoarele activităţi principale:

- Activitate I.1 (A1.1): identificarea cerinţelor utilizatorului şi analiza acceptabilităţii,
- Activitate I.2 (A1.2): investigarea fundamentelor teoretice şi a posibilităţilor de implementare,
- Activitate I.3 (A1.3): elaborarea unui cadru arhitectural standardizat comun,
- Activitate I.4 (A1.4): analiza posibilităţilor de interfaţare,

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 arhitecturii de control a proceselor şi, respectiv, analiza posibilităţilor de interfaţare.

## 2. Cerinţele utilizatorului şi analiza acceptabilităţii

### 2.1. Situaţia pe plan mondial

În prezent industria se bazează tot mai mult pe sistemele de control automat, acestea îndeplinind de la funcţii simple, precum monitorizarea proceselor şi comenzi standard, la regulatoare PID şi chiar aplicaţii complexe de control bazate pe inteligenţă artificială, metode de control predictiv sau analiza de către un supervizor a funcţionării sistemelor automate. În plus, tendinţa este de integrare a sistemelor de control ale unei instalaţii în reţelele corporatiste astfel încât să se poată aborda o metodă de management unitar, care să înglobeze aspectele economice, tehnologia de producţie, precum şi resursele materiale şi umane [4].

În ultimii ani s-au facut progrese semnificative în automatizarea instalaţiilor industriale de mari dimensiuni. Acest lucru se datorează faptului că au fost dezvoltate echipamente capabile să implementeze funcţii cu un nivel de complexitate din ce în ce mai ridicat, dar care necesită şi o ajustare corespunzătoare a mecanismelor şi strategiilor de operare şi control. Pentru aceasta, inginerii care construiesc aceste strategii trebuie să

găsească metode pentru creşterea calităţii şi eficienţei aplicaţiilor elaborate, dar şi pentru micşorarea timpului de dezvoltare şi implementare, menţinând totodată un nivel ridicat de fiabilitate şi siguranţă.

Noua generaţie de echipamente de proces trebuie să poată susţine dezvoltarea unor aplicaţii din ce în ce mai complexe şi să permită proiectarea unor sisteme de control adaptabile, (auto-)reconfigurabile, capabile să funcţioneze în arhitecturi distribuite ce au o flexibilitate ridicată [5]. Reconfigurabilitatea presupune posibilitatea de modificare a codului unei aplicaţii sau de transfer a valorilor unor variabile în timpul execuţiei, fără a fi necesară oprirea controllerului [6-8].

Trecerea de la controlul PLC (Programmable Logic Controller) centralizat la cel distribuit ajută la constituirea unui sistem capabil să se adapteze şi să fie reconfigurat în funcţie de instalaţie, care să poată fi utilizat în instalaţii distincte cu o gamă largă de controllere. Prin aceasta se creşte şi fiabilitatea sistemului întrucât nu există un punct unic de defectare şi se pot implementa mecanisme care să permită ca la înlăturarea unei componente, restul sistemului să se adapteze [8]. Principalul aspect care trebuie avut în vedere în dezvoltarea unor aplicaţii de control pentru procese complexe este posibilitatea de reutilizare a strategiilor şi algoritmilor dezvoltaţi.

La nivel mondial, evoluţia sistemelor de conducere este marcată de câteva tendinţe semnificative, dintre care menţionăm:

- Utilizarea unor standarde internaţionale atât pentru certificarea echipamentelor, a comunicaţiei sau a pachetelor de programe, cât şi pentru sisteme sau ansambluri precum bucle de reglare. Un exemplu este referitor la Sisteme cu Siguranţă Mărită (SIS − Safety Instrumented Systems).
- Diversificarea configuraţiei sistemelor, la sistemele DCS (Distributed Control Systems) şi PLC adăugându-se cele total distribuite (compatibile standardului Fieldbus Foundation), sau cele compacte PAC (Programmable Automation Controller).
- Creşterea gradului de integrare atât prin utilizarea de standarde de interfaţare, cât şi prin dezvoltarea de sisteme de conducere şi cu siguranţă mărită, bazate pe platforme de comunicaţie unice (magistrală internă).
- Extinderea utilizării de protocoale de comunicaţie standardizate pentru dezvoltarea de noi produse (traductoare, elemente de execuţie, regulatoare), cât şi de noi sisteme (DCS, PLC, RTU, SCADA, MES, PI). Menţionăm standardul IEC 61850, a cărui utilizare a creat premizele trecerii de la reţele de utilităţi monitorizate de sisteme informaţionale la Smart Grid-uri.
- Consolidarea utilizării de standarde pentru algoritmi sau expresii matematice, atât pentru automate programabile, cât şi pentru alte regulatoare sau programe de monitorizare şi control.
- Cresterea ponderii iniţiativelor şi produselor „open source” atât în sistemele informatice de gestiune, cât şi în cele de timp real. Un exemplu elocvent este iniţiativa OPC.

În ultimii 10 ani a fost introdusă în industria de proces o largă varietate de tehnologii şi echipamente de magistrale de câmp (fieldbus) ), care se bazează pe protocoale industriale precum Profibus [9], Modbus [10], Ethernet/IP [11], CAN [12], ControlNET [13] etc., ce permit schimbul de mesaje sau de date între echipamentele componente. A fost acceptat gradual faptul că sunt necesare variate tehnologii de comunicaţie pentru a răspunde la cerinţele diverselor aplicaţii. Totuşi, inginerii au nevoie de o interfaţă şi de

funcţionalitate comune pentru a opera sistemele de control, independent de tehnologia utilizată sau de producătorul acestuia. O evoluţie importantă o reprezintă standardul IEC 61804 [14], care oferă utilizatorului final specificaţiile necesare pentru satisfacerea cerinţelor sistemelor de control distribuit bazate pe blocuri funcţionale. Specificaţiile definesc cerinţele pentru ca blocurile funcţionale să controleze şi să faciliteze operaţiile de mentenanţă şi de management tehnic, ca aplicaţii care interacţionează cu elemente de execuţie şi cu echipamente de măsură. Se doreşte astfel definirea unei arhitecturi care să ofere o specificaţie comună tuturor componentelor din sistem (funcţii, echipamente, formatul datelor, metode de interfaţare etc.), precum şi a relaţiilor dintre acestea. Standardul include: modelul care defineşte componentele unui echipament compatibil cu IEC 61804; specificaţiile conceptuale ale blocurilor funcţionale de măsurare, acţionare şi prelucrare; tehnologia EDD (Electronic Device Description), care permite integrarea de detalii ale componentelor sistemului de automatizare prin utilizarea unei semantici similare instrumentelor de gestionare a ciclului de viaţă al unui sistem de control.

O aplicaţie bazată pe blocuri funcţionale este construită din componente. Blocurile funcţionale sunt încapsulări de variabile, parametri şi algoritmi de calcul, necesari în proiectarea procesului şi a sistemului său de control. Aplicaţia poate fi distribuită pe mai multe echipamente, conectate printr-o reţea sau o ierarhie de reţele de comunicaţie. În Fig. 1 sunt prezentate diverse blocuri funcţionale care se regăsesc în schema de control a unui proces industrial. Blocul tehnologic reprezintă procesul specific ataşat unui echipament. Blocul este compus din componente de achiziţie şi transformare prin care se efectuează măsurătorile sau se execută comenzile de acţionare. Blocul operaţional este specific aplicaţiei şi execută operaţii de prelucrare de semnal, cum ar fi: scalare, detectare alarme, control şi calcul. Blocul de funcţii elementare execută funcţii logice şi matematice. Blocul echipament reprezintă resursa care conţine informaţii despre echipament, despre sistemul de operare al acestuia şi despre unităţile hardware asociate.

## 2.2. Stadiul industriei în ţară

Industria sistemelor de automatizare este structurata pe categorii de companii:

- Firme producătoare de aparatură şi echipamente

Firmele producătoare de aparatură de automatizare sunt în general firme de dimensiuni mici, desprinse din personalul fabricilor cu tradiţie, precum FEA, FEPA, ITRD, IAMC, AMPLO, şi care realizează în general senzori şi traductoare de complexitate medie. Dintre întreprinderile cu tradiţie, FEPA este singura care mai supravieţuieşte la un nivel scăzut de tehnologie şi cu o paletă redusă de produse. Firmele producătoare de echipamente au fost majoritatea relocate, reduse sau înlocuite cu firme private care au preluat personalul de calitate sau protofoliul de produse.

Marile firme internaţionale stabilite în ţară au ca principal obiectiv vânzarea produselor proprii în detrimentul producţiei interne. Se profilează o schimbare în acest sens în perioada următoare prin investiţii în mici unităţi de producţie.

- Integratori de sisteme

Integratorii de sisteme sunt singura categorie care a evoluat continuu acoperind în acest moment atât realizarea de aplicaţii „la cheie”, cât şi realizarea de actualizări, modernizări sau mentenanţă. Cu toate că dispun de o putere limitată din punct de vedere

financiar şi numeric, aceasta le permite să preia o mare categorie de aplicaţii şi sisteme mici şi chiar medii în detrimentul marilor corporaţii (precum Siemens, Honeywell, Yokogawa, Emerson, Schneider).

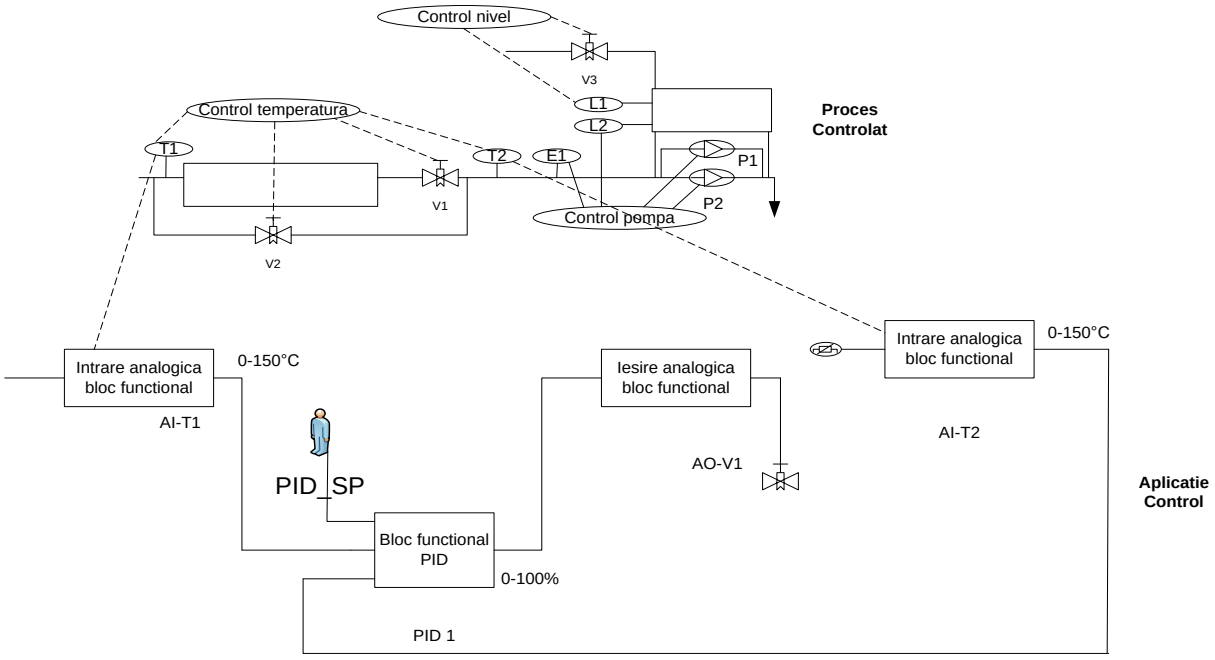



Fig. 1. Tipuri de blocuri funcţionale utilizate în conducerea proceselor

## 2.3. Cerinţele şi aşteptările clienţilor industriali

Sistemele de control bazate pe utilizarea Internetului, IBCS (Internet-Based Control System), devin din ce în ce mai populare datorită flexibilităţii şi accesibilităţii pe care le oferă. Noua generaţie de aplicaţii trebuie să se conformeze cerinţelor industriale de a susţine integrarea la nivel de companie, controlul la distanţă şi chiar implementarea de sisteme bazate pe Internet, care să permită execuţia distribuită şi cooperarea între diferite echipamente [15,16]. O altă direcţie de cercetare din domeniul sistemelor de control la distanţă de tip supervizor o reprezintă conectarea instalaţiilor la un centru dedicat de analiză şi diagnoză a riscului, capabil să ofere soluţii de gestionare a unor situaţii critice [17]. Aceasta necesită capabilităţi de execuţie la distanţă a unor algoritmi de analiză în scopul identificării situaţiilor critice, precum şi a unor algoritmi de management al riscului. Execuţia la distanţă permite unui controller dintr-o reţea distribuită să aibă acces la o putere de calcul şi o bază de cunoştinţe mai mari decât cele disponibile prin propriile resurse. În acest mod, algoritmii ce necesită un efort de calcul ridicat (din domenii precum modelarea proceselor, optimizare, control avansat bazat pe tehnici de inteligenţă artificială, analiză a riscului, prelucrare de imagini etc.) pot fi stocaţi şi executaţi pe un echipament aflat la distanţă, folosind o conexiune de reţea pentru a accesa parametrii din proces şi a trimite rezultatele.

Se acordă o atenţie tot mai mare dezvoltării de servicii cloud dedicate, care să preia atribuţiile actualelor sisteme de conducere SCADA (Supervisory Control and Data Aquisition) şi să pună la dispoziţie funcţionalitatea acestora pentru a construi arhitecturi dinamice ce acţionează şi pe planul vertical, intregrând datele din proces cu sisteme precum MES (Manufacturing Execution System) şi/sau ERP (Enterprise Resource

Planning [18]. Aceste servicii permit relocarea resurselor de prelucrare în afara organizaţiei, oferind putere de calcul şi spaţiu de stocare superioare celor disponibile într-o instalaţie, capabile să asigure atât nivele mari de „uptime", precum şi redundanţa, siguranţa şi integritatea datelor [19]. Toate acestea se aliniază direcţiei internaţionale de dezvoltare şi interconectare a tot mai multor echipamente, cunoscută şi sub numele de Internet of Things – IoT [20,21]. Această tendinţă are ca obiectiv adăugarea de module inteligente fiecărui echipament care să îi permită accesul la o gamă mai mare de informaţii şi servicii (management al activelor, planificarea resurselor, execuţie la distanţă, control autonom şi/sau distribuit etc.), precum şi distribuirea datelor către alte dispozitive din reţea. Principalele elemente ale IoT sunt reprezentate de senzorii încorporaţi (embedded), tehnologiile de recunoaştere a imaginilor, cu aplicaţii atât industriale cât şi pentru consumatori, şi extinderea tehnologiei de comunicaţie NFC (Near Field Communication) pentru aplicaţii de eficientizare a serviciilor, precum efectuarea plăţilor [5].

Cloud Computing şi Internet of Things se regăsesc între tendinţele prezentate la simpozionul Gartner pentru identificarea tehnologiilor strategice pentru 2012 [5], alături de:

- dezvoltarea tehnologiilor mobile şi a aplicaţiilor asociate acestora;
- integrarea experienţei utilizatorilor în modul de prezentare a informaţiei astfel încât să se anticipeze nevoile acestuia;
- analiza datelor din surse multiple, în timp real, pentru a putea simula şi prezice comportamentul viitor al unui sistem, eventual cu utilizarea unor resurse de tip cloud;
- identificarea unor soluţii pentru gestionarea unor structuri de date de mari dimensiuni.

Din punctul de vedere al sistemelor de automatizare şi control, principalele tendinţe identificate pentru viitor sunt [22]:

- implementarea unor sisteme de control care să uniformizeze fluxul de operaţii şi informaţii la nivel de întreprindere;
- simplificarea arhitecturilor de control prin contopirea mai multor nivele, astfel ca informaţia să poată ajunge, de exemplu, direct de la nivelul 0 al echipamentelor de proces la nivele superioare precum MES sau ERP;
- creşterea performanţelor echipamentelor de proces;
- adoptarea unor standarde de comunicaţie bazate pe servicii web, precum OPC UA;
- definirea unor instrumente pentru analiza unor seturi mari de date;
- dezvoltarea de module software sau echipamente care să permită adăugarea de noi funcţionalităţi instalaţiilor existente fără a renunţa la vechile componente;
- extinderea capabilităţilor de monitorizare la distanţă prin dezvoltarea de aplicaţii mobile;
- dezvoltarea de soluţii avansate pentru securitatea informaţiei, de exemplu prin migrarea la IPv6;
- definirea unor strategii de proiectare integrate care să includă instrumente de modelare şi optimizare pentru obţinerea unor procese mai eficiente.

Alte cerinţe importante sunt detaliate în continuare.

*Integrarea unor metode de detecţie a avariilor şi diagnoză şi a unor tehnici de analiză a pericolelor pentru a efectua estimări online de risc şi monitorizare***.** Lucrările în domeniul diagnozei avariilor au relevat faptul că sistemele tehnice de diagnoză ar trebui să efectueze şi estimarea riscului; această abordare a fost neglijată în special datorită naturii

aplicaţiilor ţintă. Analiza online a riscului implică operaţii complexe statistice, bazate pe o cantitate mare de date, ce nu au putut fi efectuate în maniera clasică, cum ar fi pentru aplicaţii în domeniul aeronautic, unde este necesar un timp de reacţie foarte scurt în cazul în care apare o defecţiune. În ceea ce priveşte aplicaţiile industriale, este necesară stabilirea stării defecţiunii, dacă a fost sau nu semnalată înainte, dacă este tolerabilă, tolerabilă cu precauţii, sau intolerabilă.

*Stabilirea strategiilor de prevenire a pericolelor şi de închidere în siguranţă a instalaţiei atunci când nivelul riscului este considerat a fi prea ridicat*. Aşa cum s-a arătat şi în cele mai recente studii [17, 23], managerii de instalaţii s-au confruntat cu dificultăţi în pregătirea operatorilor de proces, referitoare la acţiunile pe care ei ar trebui să le ia în caz de urgenţă. De asemenea, există situaţii când riscul de apariţie a unei situaţii periculoase nu este evaluat corect şi apar întârzieri în închiderea instalaţiei, ceea ce poate pune probleme serioase, chiar ameninţă siguranţa populaţiei în unele cazuri. Fiecare proces are propriile particularităţi privind închiderea instalaţiei, de care operatorii trebuie să fie conştienţi. De aceea este imposibil de implementat o procedură sigură de închidere care să funcţioneze corect pentru fiecare aplicaţie. Totuşi, pot fi implementate alte câteva proceduri generice pentru diferite tipuri de instalaţii, ce pot constitui linii directoare pe care să le urmeze operatorii într-o situaţie de urgenţă.

*Analiza legăturii puternice ce există între prevenirea defecţiunilor şi problemele de optimizare a sistemului pentru a obţine rezultate corecte şi concludente.* Optimizarea sistemului şi siguranţa operaţională au fost tratate ca două probleme separate pentru o lungă perioadă de timp [24, 25]. Un sistem de diagnoză tehnică poate contribui la optimizarea sistemului prin prelucrarea noilor limite admisibile ale domeniului după apariţia unei/unor avarii şi transmiterea lor către componenta responsabilă de optimizarea rezolvării problemei, ca date de intrare pentru problemă. Pe de altă parte, modulul de optimizare a sistemului joacă un rol important în prevenirea defecţiunilor prin calcularea referinţelor optimale pentru parametrii controlaţi, conducând astfel la minimizarea criteriilor utilizate pentru a descrie distanţa dintre comportamentul normal şi post-avarie a sistemului.

## 2.4. Standarde utilizate

Principalele piedici în reutilizarea unor secţiuni ale logicii de control dintr-o aplicaţie sunt lipsa unei standardizări a funcţiilor bloc şi/sau interdependenţa între echipamentul hardware şi modulele de program. Standardul IEC 61131 [26] a fost dezvoltat pentru rezolvarea acestei probleme, însă lipsa unor specificaţii referitoare la ordinea de execuţie a blocurilor şi diferenţele de reprezentare la nivel de limbaj maşină au făcut ca reutilizabilitatea să nu fie posibilă.

Standardul IEC 61499 [27] fost creat pornind de la caracteristicile lui IEC 61131, pentru a susţine cerinţele de reutilizabilitate, reconfigurabilitate şi adaptabilitate utilizând o metodă de programare bazată pe funcţii bloc. Acest standard oferă o nouă abordare din punctul de vedere al reutilizabilităţii. O aplicaţie se construieşte ca un tot, implementarea fiind bazată pe funcţionalitate şi nu pe echipamente. Aplicaţia se poate diviza apoi în subaplicaţii, fiecare putând fi asociată unui anumit echipament, corespunzător acelei interfeţe. Adoptarea acestui standard la nivel industrial a început cu procesele de fabricaţie [28], însă există şi studii care au analizat posibilitatea de utilizare pentru aplicaţii de

control al proceselor de tip batch [29-31] şi chiar pentru control în buclă închisă [32,33]. Pe măsură ce standardul câştigă tot mai mulţi adepţi, se doreşte lărgirea domeniului de aplicaţie prin dezvoltarea unor sisteme care să permită integrarea web a acestuia.

Reglementarea din standard prin intermediul unor profiluri de conformitate duce la obţinerea unor instrumente software deschise din punctul de vedere al portabilităţii, configurabilităţii şi interoperabilitaţii, asigurând astfel un suport ridicat pentru reutilizabilitatea funcţiilor bloc dintr-o aplicaţie [34]. De asemenea, trecerea de la controlul PLC centralizat la cel distribuit permite constituirea unui sistem capabil să se adapteze şi să fie reconfigurat în funcţie de instalaţie, care să poată fi utilizat în instalaţii distincte cu o gamă largă de regulatoare. Din punctul de vedere al arhitecturii [35], sistemele de control pot fi: sisteme cu control centralizat (ca majoritatea automatelor de tip PLC existente, ce utilizează standardul IEC 61131), sisteme cu control ierarhizat (unde apar modelele de intrări-ieşiri distribuite, întâlnite atât la DCS-uri, cât şi la PLC-uri) şi sisteme cu control distribuit (unde funcţiile de control sunt distribuite între echipamentele care formează aplicaţia şi care lucrează împreună pentru îndeplinirea cerinţelor). Cel mai performant sistem distribuit se bazează pe reglementările tehnologiei Fieldbus Foundation şi este promotorul standardului IEC 61104. Standardul IEC 61499 aduce noutatea distribuirii unei aplicaţii software între mai multe echipamente care pot acţiona independent pentru a-şi îndeplini funcţiile [27]. Se pot implementa astfel arhitecturi bazate pe agenţi inteligenţi, capabili să lucreze împreună în sisteme distribuite sau să se reconfigureze automat pentru asigurarea unor arhitecturi deschise, modulare, scalabile şi tolerante la defecte.

O analiză asupra diferenţelor de abordare şi a performanţelor obţinute în proiectarea unor sisteme distribuite bazate pe IEC 61131 şi respectiv pe IEC 61499 a fost făcută în [36]. Sunt analizate aspecte referitoare la uşurinţa de implementare, la instrumentele disponibile, la dimensiunea şi funcţionalitatea sistemului, precum şi la caracteristicile hardware ale echipamentelor implicate şi este propusă o metodologie de evaluare a conformităţii şi performanţelor obţinute în urma utilizării celor două standarde într-o aplicaţie distribuită. Se arată că pentru majoritatea criteriilor se obţin rezultate mai bune prin utilizarea standardului IEC 61499, avantajele lui IEC 61131 bazându-se în special pe gama mai mare de echipamente disponibile şi pe experienţa inginerilor de proces care nu necesită astfel instruire suplimentară.

OPC UA este cea mai recentă specificaţie OLE pentru Controlul Proceselor (OPC) de la OPC Foundation şi diferă semnificativ de predecesorii săi prin faptul că include capabilităţi ridicate de asigurare a securităţii şi fiabilităţii datelor, permite integrarea modelelor de date, precum date istorice, alarme sau programe şi nu mai este dependent de utilizarea DCOM din Windows, ceea ce oferă o mai mare flexibilitate. Open Platform Communications Unified Architecture sau OPC UA este un standard de comunicaţie ce conţine un set de documente care oferă reguli şi informaţii despre modul cum aplicaţiile software şi echipamentele pot trimite şi primi diferite tipuri de date. Scopul standardului este de a oferi o cale prin care aplicaţiile software să comunice cu diferite tipuri de automate programabile şi echipamente de proces fără a fi necesară o implementare a unui sistem software dedicat, comercial, prin utilizarea unei platforme independente şi sigure.

Arhitectura OPC este o arhitectură tipică client-server (vezi Fig. 2). Utilizarea stivelor client şi server, dar şi a interfeţei API (Application Programming Interface) asigură flexibilitatea şi posibilitatea de a unifica diverse tipuri de servere, permiţând integrarea sistemelor de control cu alte aplicaţii şi făcând posibilă migrarea de la aplicaţiile curente la

cele bazate pe OPC UA. Se pot integra echipamente de la diverşi producători, datele putând fi vizualizate într-o interfaţă unică, pentru a putea fi încoporate în sisteme ERP sau pentru a fi accesate din aplicaţii de control bazate pe medii de dezvoltare ale altor producători. Aceasta a dus la o mare popularitate a standardului, în prezent existând interfeţe OPC pentru conectarea la peste 150 echipamente de la cei mai cunoscuţi furnizori, precum şi interfeţe specifice protocoalelor de comunicaţie industriale (Profibus, Modbus, DNP, BACNet, CAN etc.) şi soluţii pentru asigurarea redundanţei bazate pe acest standard [37,38].

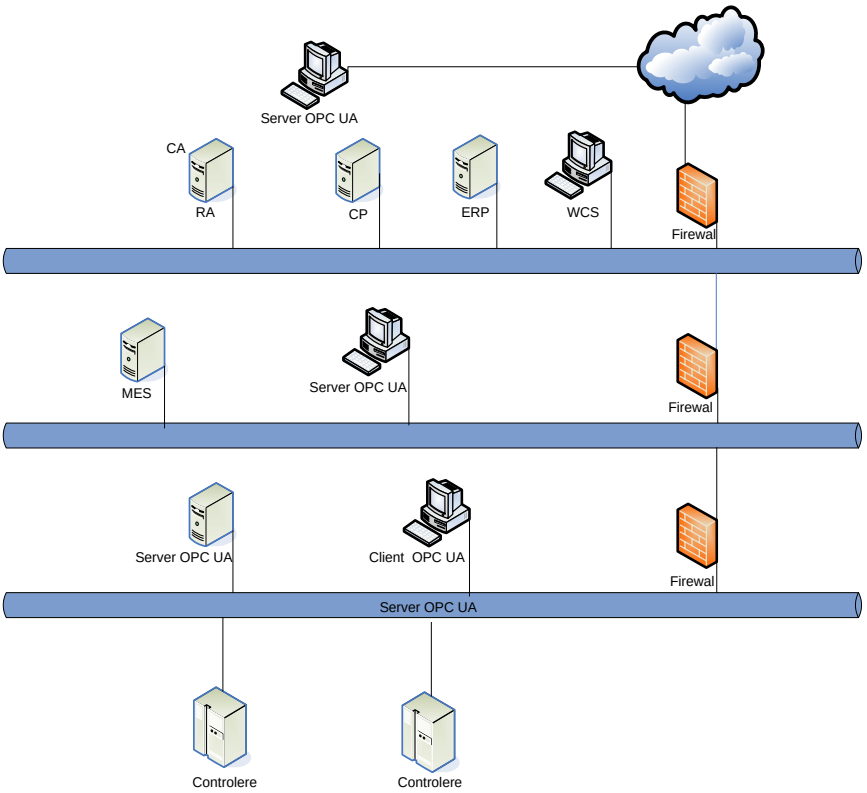



Fig. 2. Arhitectura client-server OPC

## 2.5. Funcţii bloc şi elaborarea aplicaţiilor de control

În dezvoltarea unei aplicaţii de control, inginerii încearcă să identifice cele mai simple metode de programare ce le permit implementarea unor sisteme din ce în ce mai complexe. Echipamentele actuale pentru controlul proceselor, de tipul DCS sau PLC, au apărut în jurul anilor 1960 [5], ca o necesitate în evoluţia de la sistemele de control bazate pe relee şi circuite analogice. În cazul PLC-urilor, metoda de programare a fost reprezentată la început de diagramele Ladder, datorită asemănării cu schemele logice cu relee, sau de listele de instrucţiuni în limbaj de asamblare, care erau mai apropiate de modul de programare a calculatoarelor din acea vreme. În prezent, cea mai des întâlnită metodă pentru programarea PLC-urilor este standardul IEC61131-3 [26], care defineşte cinci limbaje de programare pentru acestea: diagrame de funcţii bloc (FBD – Function Block Diagram), diagrame ladder (LD – Ladder Diagram), text structurat (ST – Structured Text), listă de instrucţiuni (IL – Instruction List) şi funcţii secvenţiale (SFC – Sequential Flow Chart). Un avantaj important este faptul că standardul permite utilizarea mai multor limbaje de programare în cadrul aceluiaşi controller.

Funcţiile bloc reprezintă o metodă de programare bazată pe simboluri grafice în care se pune în evidenţă fluxul de date de la intrare la ieşire, după cum se poate observa în Fig. 3. Ele sunt blocuri „drag-and-drop" disponibile în bibliotecile de programe de dezvoltare a aplicaţiilor de control (precum STEP 7, ISaGRAF, FBDK, NxtLIB etc.) ce pot fi interconectate prin trasarea grafică de conexiuni între variabilele de ieşire şi de intrare a diferitelor funcţii bloc. Aceste conexiuni simbolizează traseul fluxului de date.

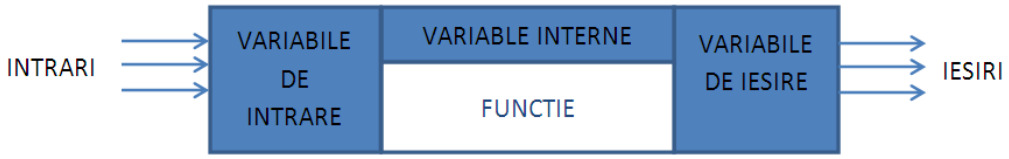



Fig. 3. Modelul general al unei funcţii bloc

Modelul general al unei funcţii bloc constă în variabile de intrare, variabile de ieşire, variabile interne şi funcţia implementată. Variabilele de intrare pot fi doar citite în interiorul blocului, iar cele interne şi cele de ieşire pot fi şi citite şi scrise. Execuţia unei funcţii bloc poate fi continuă (Simulink), discretă (IEC 61131-3), sau bazată pe evenimente (IEC 61499) [39,40].

Cei mai cunoscuţi fabricanţi de sisteme de automatizare (Siemens, Allen Bradley, Fanuc, Wago, Eaton, Emerson, Yokogawa, Honeywell etc.), de la cele mai simple la cele mai complicate, oferă posibilitatea de programare a acestora utilizând funcţii bloc, fie bazate pe standardul IEC 61131-3, fie specifice mediului de programare propriu (comercial). Motivul este faptul că funcţiile bloc reprezintă o metodă intuitivă de realizare a logicii de control care nu necesită cunoştinţe avansate de programare. Principalele avantaje în utilizarea funcţiilor bloc în controlul proceselor sunt:

- Posibilitatea de încapsulare a funcţiilor, făcând programul mai uşor de urmărit;
- Posibilitatea de reutilizare a unei funcţii prin construirea de noi instanţe cu variabile de intrare sau de ieşire diferite;
- Posibilitatea de construire a unei biblioteci de funcţii, a cărei utilizare va micşora timpul de dezvoltare a unor aplicaţii viitoare;
- Uşurinţa în efectuarea de modificări, deoarece prin modificarea unei funcţii bloc se vor modifica toate instanţele acesteia.

Cele mai utilizate metode de reprezentare bazate pe funcţii bloc sunt IEC 61131, IEC 61499 şi Simulink.

Simulink este parte a mediului de programare MATLAB, utilizat la scară largă în special în mediul academic, care dispune de o bibliotecă cu un număr mare de funcţii de modelare, simulare şi optimizare. Simulink este folosit de cele mai multe ori pentru dezvoltarea unor modele matematice ale procesului şi simularea unor strategii complexe de control (predictiv, bazat pe predictori Smith, etc.), fără a utiliza o legătură în timp real la proces. Utilizarea acestui mod de programare în controlul proceselor a fost posibilă odată cu crearea modulelor de compilare şi conversie în standarde precum IEC 61131-3, însă aplicaţii concrete care să utilizeze acest mediu de dezvoltare pentru realizarea şi implementarea programelor de control uzuale nu sunt numeroase, probabil datorită costului ridicat şi al complexităţii crescute.

Standardul IEC 61131-3 [26] este în prezent cea mai utilizată metodă pentru programarea echipamentelor de tip PLC. Chiar dacă nu asigură portabilitatea codului între diferite aplicaţii, aşa cum se dorea iniţial, acest standard are numeroase avantaje precum: programare standardizată (ceea ce implică timp mai redus de dezvoltare şi de depanare comparativ cu soluţiile proprietare), modularitate, posibilitatea de utilizare a mai multor modalităţi de programare în aceeaşi aplicaţie (diagrame ladder, listă de instrucţiuni, text structurat sau funcţii bloc), încapsulare, posibilitatea de utilizare a unor funcţii deja existente în aplicaţia de dezvoltare, suport pentru controlul execuţiei etc. Din punctul de vedere al utilizării în controlul proceselor, standardul IEC 61131-3 deţine o pondere semnificativă a aplicaţiilor dezvoltate până în prezent, în toate domeniile asociate (linii de fabricaţie, instalaţii industriale de control, sisteme încorporate, automatizări de clădiri, reţele de apă şi canalizare, sisteme energetice etc.). Acest lucru se datorează multitudinii de echipamente hardware care suportă acest standard, a aplicaţiilor de dezvoltare utilizate pentru programarea lor şi a simplităţii dezvoltării unor aplicaţii complexe prin accesul la biblioteci de funcţii de control, module de conectare hardware, interfeţe de comunicaţie, simboluri grafice etc.

Standardul IEC 61499 [27,39] a fost dezvoltat ca o îmbunătăţire pentru IEC 61131-3, păstrând avantajele acestuia şi adaugând funcţionalităţi suplimentare pentru crearea unor aplicaţii distribuite, reutilizarea unui cod în aplicaţii diverse şi folosind medii de programare diverse, controlul fluxului de execuţie prin intermediul unor semnale de tip eveniment, definirea unor funcţii bloc de tip interfaţă, şi dezvoltarea aplicaţiilor independent de platforma hardware. Datorită aspectelor legate de siguranţă şi securitate, care au apărut în lipsa unor practici fundamentate de elaborare, există destul de puţine aplicaţii industriale concrete bazate pe acest standard. Până în prezent au fost dezvoltate câteva aplicaţii menite să pună în evidenţă beneficiile standardului, cele mai importante fiind o fabrică de procesare a cărnii din Noua Zeelandă, o linie experimentală pentru fabricarea pantofilor, un sistem de sortare a bagajelor şi câteva sisteme de control şi automatizare pentru clădiri [5,41].

În domeniul proceselor de fabricaţie se observă o tendinţă a clienţilor de a avea cerinţe specifice în realizarea unui produs [42]. Aceasta implică modificarea aplicaţiei şi creşterea timpului de implementare a unui sistem. Se doreşte tot mai mult o abordare orientată pe componente care să permită integrarea operaţiunilor de preverificare şi validare a unor secţiuni reutilizabile dintr-o aplicaţie într-un mediul simulat şi comutarea ulterioară pentru integrarea în procesul real. Prin utilizarea unor module software predefinite se pot reduce timpul total de punere în funcţiune, precum şi costurile asociate.

Reutilizabilitatea se referă la posibilitatea utilizării unei funcţii bloc care implementează un anumit algoritm în diferite aplicaţii, fără a fi constrânsă de echipamentul pe care această funcţie va fi implementată. Acest concept permite, pe de o parte, micşorarea efortului şi a timpului de dezvoltare a unei aplicaţii, şi, pe de altă parte, oferă o nouă modalitate de abordare a aplicaţiilor distribute, prin faptul că se poate gândi aplicaţia ca un întreg, pentru ca ulterior aceasta să fie divizată în subaplicaţii ce pot fi implementate pe diferite echipamente. Astfel, se permite dezvoltarea unor aplicaţii complexe bazate pe arhitecturi flexibile într-un timp mai scurt şi cu o calitate ridicată, ceea ce implică costuri reduse. De asemenea, este important ca funcţia bloc sa fie utilizată în aplicaţie ca instanţă, astfel ca o modificare a funcţiei în bibliotecă să se regăsească în toate instanţele acesteia din program.

Principalele piedici în reutilizarea unor secţiuni ale logicii de control dintr-o aplicaţie sunt lipsa unei standardizări a funcţiilor bloc şi/sau interdependenţa între echipamentul hardware şi modulele de program. Acest neajuns se întâlneşte şi în cazul mediului MATLAB Simulink, întrucât acesta se bazează pe un format propriu, proprietar. Totuşi, modulul PLC coder (de la MATLAB) sau aplicaţia PLC Link de la DEIF [43] permit generarea automată a codului bazată pe standardul IEC 61131-3 şi utilizarea acestuia în cele mai cunoscute medii de dezvoltare pentru PLC-uri (CoDeSys, Step 7, RsLogix). Chiar şi aşa, există foarte puţine referinţe ale folosirii acestor instrumente în aplicaţii de timp real pentru controlul proceselor.

În cazul aplicaţiilor dezvoltate pe baza standardului IEC 61131-3, funcţiile bloc folosesc varibile globale care reprezintă interfeţele hardware. Atât funcţiile, cât şi aceste variabile, sunt apelate la fiecare ciclu de procesor. Acest ciclu depinde de echipamentul hardware pe care este rulată aplicaţia şi are o influenţă mare asupra timpului total de execuţie. De asemenea, mediile de dezvoltare pentru IEC 61131-3 folosesc metode diferite pentru compilarea şi stocarea funcţiilor bloc în biblioteca proprie. Astfel, nu există o compatibilizare care să permită utilizarea unei funcţii bloc bazate pe acest standard dintr-un mediu de dezvoltare în altul. Altfel spus, echipamentele care funcţionează pe acest standard pot fi programate cu mediile de dezvoltare proprietare, caz în care reutilizabilitatea este la fel de limitată ca la DCS-uri. Totuşi, se poate obţine un anumit grad de reutilizare prin intermediul unor aplicaţii de dezvoltare precum CoDeSys [44] sau ISaGRAF [45] care includ drivere la mai multe echipamente, ceea ce permite dezvoltarea unor aplicaţii comune ce pot fi ulterior compilate pentru echipamente hardware specifice.

Pentru crearea de cod reutilizabil este necesară decuplarea în programe a interfeţelor hardware încorporând logica de control şi a punctelor de intrare/ieşire. Standardul IEC 61499 oferă suportul necesar pentru astfel de secvenţe de control prin definirea unor funcţii bloc de tip interfaţă SIFB (Service Interface Function Block) ce se pot conecta la funcţiile bloc simple sau compuse. Acestea sunt nişte module de intrări-ieşiri ce pot conecta intrările la un echipament doar pe baza unui identificator al tipului echipamentului, fără a mai fi astfel necesară modificarea unei întregi funcţii bloc de interfaţare. Totuşi, nu există limitări din punctul de vedere al locaţiei unde se pot folosi aceste interfeţe, astfel că ele pot apărea în interiorul unei funcţii compuse, anulându-i astfel caracterul reutilizabil. Din acest motiv, o practică obligatorie este să nu se folosească SIFB în interiorul unei funcţii compuse, chiar dacă aceasta poate presupune permiterea accesului la toate intrările şi ieşirile la cel mai înalt nivel de încapsulare a aplicaţiei. Chiar şi aşa, utilizarea unor SIFB-uri face ca aplicaţia în ansamblu să fie dependentă de echipamentul hardware.

În încercarea de a soluţiona această problemă, au fost dezvoltate interfeţe SIFB generice (pentru intrări şi ieşiri analogice şi digitale), care includ specificaţiile mai multor echipamente hardware, putând fi astfel utilizate pe platforme multiple [42,46]. Această soluţie a fost demonstrată prin utilizarea mediului de dezvoltare FORTE, care avea implementate specificaţiile pentru fiecare dintre platformele hardware testate. Din acest motiv, această soluţie nu mai este valabilă la utilizarea unui alt mediu de programare.

În [47] se propune o altă abordare, prin utilizarea unei metode de proiectare specifice şi aplicarea conceptului de interfeţe adaptor definite pentru standardul IEC 61499. Aceste interfeţe au avantajul că separă reţeaua de funcţii bloc care constituie aplicaţia în două secvenţe: de control şi de interfaţare (intrări/ieşiri). Mai mult, secvenţa de

interfaţare va conţine numai interacţiunea cu echipamentele hardware, procesarea datelor primite făcându-se tot la nivelul de control.

Metoda de proiectare a sistemului de control prezentată în [47] presupune respectarea unor principii de bază:

- utilizarea explicită a sincronizării prin evenimente;
- posibilitatea de identificare uşoară a interfeţelor de proces;
- separarea aplicaţiei de control de logica de interfaţare;
- organizarea ierarhică a aplicaţiilor de control în subcomponente, reprezentate prin funcţii bloc compuse sau subaplicaţii şi legate la proces prin interfeţe adaptor.

Din punctul de vedere al compatibilităţii între instrumentele software ce pot fi folosite în programarea aplicaţiilor bazate pe acest standard, în [34] este făcută o evaluare între diverse instrumente software de dezvoltare şi execuţie, după cum se poate vedea şi în tabelele de mai jos. Dintre acestea, FBDK (Function Block Development Kit) [48] este primul mediu de dezvoltare gratuit care a fost creat iniţial pentru a demonstra caracteristicile şi beneficiile standardului IEC 61499 şi ulterior a fost îmbunătăţit continuu pentru a reflecta şi testa schimbările propuse asupra acestui standard [34].

***Tabelul 1. Portabilitatea elementelor bibliotecii între diferite medii software [34]***

| | FBDK | 4DIAC-IDE | nxtSTUDIO | ISaGRAF Workbench |
|---|---|---|---|---|
| FBDK | x | x | x | |
| 4DIAC-IDE | x | x | x | |
| nxtSTUDIO | x | x | x | |
| ISaGRAF Workbench | | | | x |

***Tabelul 2. Configurabilitatea echipamentelor utilizând diferite medii software [34]***

| | FBDK | 4DIAC-IDE | nxtSTUDIO | ISaGRAF Workbench |
|---|---|---|---|---|
| FBRT | 1+ | 1+ | | |
| FORTE | 1+ | 1+ | | |
| nxtRT61499F | | | 2 | |
| ISaGRAF Runtime | | | | 2 |

Notaţia 1+ în Tabelul 2.2 înseamnă drepturi de scriere, citire, start, stop, reset, interogare a oricărui tip de date sau de funcţii bloc, creare/ştergere funcţii bloc, conexiuni, aplicaţii, interogare tipuri de funcţii bloc şi tipuri de date, iar 2 adaugă în plus posibilitatea creării/ştergerii tipurilor de date sau a tipurilor de funcţii bloc.

Se observă astfel că, în cazul standardului IEC 61499, reglementarea prin intermediul unor profiluri de conformitate a dus la obţinerea unor instrumente software

deschise din punctul devedere al portabilităţii, al configurabilităţii şi al interoperabilităţii, asigurând astfel un suport ridicat pentru reutilizabilitatea funcţiilor bloc dintr-o aplicaţie.

***Tabelul 3. Interoperabilitatea între echipamente [34]***

| | FBRT | FORTE | nxtRT61499F | ISaGRAF Runtime |
|---|---|---|---|---|
| FBRT | x | x | x | |
| FORTE | x | x | x | |
| nxtRT61499F | x | x | x | |
| ISaGRAF Runtime | | | | x |

## 2.6. Biblioteci de algoritmi

O bibliotecă de algoritmi constituie o colecţie de funcţii, organizate într-o manieră intuitivă, care respectă anumite standarde de documentare şi implementare. Ideea de bibliotecă de algoritmi este prezentă în toate mediile de dezvoltare de aplicaţii de control prin intermediul unui set de funcţii predefinite (funcţii matematice simple, funcţii logice, funcţii de acces la modulele de intrări/ieşiri, funcţii de temporizare, multiplexoare, bistabili etc.), care pot fi utilizate în construirea unei aplicaţii software. Acestea sunt de cele mai multe ori specifice acelui mediu de programare şi nu pot fi utilizate în afara lui.

MATLAB [49] şi LabView [50] sunt aplicaţii comerciale utilizate la scară largă în controlul proceselor, care pun la dispoziţia utilizatorului biblioteci cu un număr mare de funcţii din domenii precum modelarea proceselor, simulare, optimizare, prelucrarea semnalelor. De asemenea, SLICOT [51] este o bibliotecă disponibilă online (public pentru utilizări academice), care conţine algoritmi numerici pentru calcule de complexitate ridicată din domeniul teoriei sistemelor. Aceasta utilizează bibliotecile de rutine de algebră liniară BLAS [52] şi LAPACK [53] cu scopul de a oferi noi metode de analiză şi proiectare pentru controlul proceselor.

Pe lângă aceste biblioteci orientate mai mult către mediul academic, există şi unele orientate către implementarea practică a funcţiilor de control. Thinkcycle [54] este un astfel de exemplu. Acesta este un proiect de cercetare industrială bazat pe web, care are ca scop constituirea unei platforme colaborative deschise permiţând schimbul de informaţii între ingineri, proiectanţi, cercetători şi alţi specialişti din domenii de aplicaţie variate.

Două dintre principiile de bază ale standardului IEC 61499 sunt reutilizabilitatea şi folosirea unei metode de reprezentare deschise. Aceste principii permit constituirea unor biblioteci de algoritmi care să ofere un real suport inginerilor de control al proceselor în dezvoltarea şi implementarea unor soluţii complexe cu fiabilitate ridicată într-un timp scăzut. În acest sens în [34] este reglementat un set de profiluri de conformitate menit să asigure portabilitatea între bibliotecile specifice diferitelor medii de programare. Fiecare dintre mediile de dezvoltare de aplicaţii au biblioteci de algoritmi specifice. Dintre acestea, se remarcă cei de la NxtControl, care au lansat un produs separat, numit nxtLib, ce pune la dispoziţia utilizatorului un număr mare de funcţii de control, de HMI (Human-Machine Interface) sau interfeţe pentru componentele hardware [55]. De asemenea, aceste medii permit dezvoltarea continuă a bibliotecii prin adăugarea de noi funcţii create de utilizator. Seturi de funcţii bloc independente de aplicaţie sau de mediul de programare folosit au fost

propuse şi în unele lucrări de cercetare. În [56] şi [57] sunt prezentate astfel de biblioteci de funcţii bloc pentru controlul în buclă închisă al proceselor, respectiv pentru sisteme de măsurători industriale şi control al proceselor. Mare parte din funcţiile prezentate în [57] sunt disponibile în mediile de dezvoltare gratuite existente sau pot fi reproduse pe baza informaţiilor oferite. Din păcate, componentele bibliotecii prezentate în [56] sunt inaccesibile persoanelor din afara grupului academic care le-a dezvoltat.

Biblioteca proiectului CALCULOS va conţine elemente care nu se regăsesc în mediile de programare gratuite precum FBDK (Function Block Development Kit) [58] sau 4DIAC (Framework for Distributed Industrial Automation and Control) [59]. Astfel de medii vor fi utilizate în implementarea sistemului şi dezvoltarea algoritmilor din bibliotecă. Pentru elaborarea unor algoritmi mai complecşi este posibil să fie nevoie de extinderea caracteristicilor mediilor de programare vizate pentru a putea susţine funcţionalitatea acestora. O astfel de problematică a fost abordată în [15], unde se studiază cerinţele de implementare a unui algoritm predictiv bazat pe MPC (Model Predictive Controller) pe un echipament de control.

## 2.7. Mediul de dezvoltare ISaGRAF

ISaGRAF este un mediu de dezvoltare a aplicațiilor de control automat ce permite crearea de sisteme locale și distribuite. Acesta oferă un motor de control robust și un cadru intuitiv pentru elaborarea aplicațiilor. ISaGRAF este de asemenea un sistem de programare bazat pe logică simplă ce suportă toate limbajele de programare incluse în standardul IEC 61131-3 (LD, FBD, ST, IL, SFC), ceea ce este ideal pentru aplicații industriale cum ar fi sisteme de achiziții de date, sisteme de control distribuite, automatizarea sistemelor de construcții, controlul mișcării și sisteme de control de tip wireless. ST și IL sunt limbaje de programare textuală, iar celelalte sunt limbaje de programare grafică. De asemenea, ISaGRAF suportă standardul IEC 61499-1, permiţând îndeplinirea următoarelor cerințe:

- Portabilitatea: instrumentele software pot accepta și interpreta corect componentele software și configurațiile de sistem produse de alte instrumente software;
- Interoperabilitatea: dispozitivele încorporate pot opera împreună pentru a efectua funcțiile de care este nevoie pentru aplicațiile distribuite;
- Configurabilitatea: orice dispozitiv împreună cu componentele sale software pot fi configurate de sisteme software provenite de la furnizori multiplii.

Nucleul de rulare ISaGRAF este un motor portabil de execuție ce are comportament asemănător cu controllerele programabile tradiționale. Totuși acesta le depășește în performanţă și flexibilitate. Nucleul de rulare are de asemenea funcții suplimentare, cum ar fi controlul proceselor și controlul mișcării, ce sporesc posibilitățile aplicațiilor.

Bancul de lucru ISaGRAF este un mediu de dezvoltare a aplicațiilor folosit pentru crearea de programe a logicii de control în oricare din limbajele de programare ale standardului IEC61131-3, dedicat pentru dezvoltarea aplicațiilor de control. Aceste aplicații pot fi distribuite pe diferite platforme ce vor comunica între ele prin rețea. Un proiect ISaGRAF va expune legăturile și distribuția fiecărei bucle a unui controller programabil. Acestea sunt executate de mașina virtuală ISaGRAF pe fiecare platformă. Bancul de lucru verifică sintaxa unui cod sursă şi realizează un test de coerenţă a aplicaţiei, de pildă, corectitudinea legăturilor. De asemenea, ISaGRAF generează un cod care poate fi

simulat sau încărcat şi rulat pe maşina ţintă. Se oferă şi facilităţi de depanare şi examinare a „profilului" execuţiei, pentru optimizarea aplicaţiei.

ISaGRAF 5.2 (2009) oferă posibilitatea de a utiliza limbaje de programare grafice: Sequential Function Chart (SFC), Flow Chart (FC), Functional Block Diagram (FBD), Function Block Diagram − IEC 61499 şi Ladder Diagram (LD), cât şi limbaje de programare textuale: "Structured Text" (ST) şi "Instruction List" (IL). Primele două limbaje au editoare dedicate, în timp ce toate celelalte folosesc un editor multilingv.

Bancul de lucru ISaGRAF 6 (http://www.isagraf.com/index.htm) este un mediu modular şi flexibil care permite utilizatorilor să adauge sau elimine componente (inclusiv adăugarea de editoare specializate, cum ar fi editorul de diagrame SAMA, pentru industria energetică). Fiecare componentă a fost elaborată cu, şi interacţionează prin, noua tehnologie ISaGRAF, bazată pe cadrul Microsoft .NET®, numită „Automation Collaborative Platform" (ACP). Mediul de lucru ISaGRAF oferă editoare intuitive, grafice şi textuale, incluzând ferestre dimensionabile, operaţii „ştergere şi copiere", mutarea obiectelor etc.

Un proiect ISaGRAF este divizat în unul sau mai multe bucle PLC sau „resurse", care identifică platformele hardware şi defineşte legăturile dintre ele. Configuraţiile (adică, platformele hardware conţinând una sau mai multe resurse) şi reţelele de comunicaţii reprezintă divizarea fizică a unui proiect, în timp ce o resursă reprezintă o ţintă de execuţie de tip „maşină virtuală". Facilităţile de management a programelor permit definirea modulelor, operaţiilor şi interacţiunilor acestora. Este facilitată reutilizarea unităţilor de cod.

O „placă I/O" poate reprezenta, de pildă, plăci fizice, I/O la distanţă şi transferuri în memoria virtuală. Datele sunt declarate folosind un browser de variabile. Toate datele unui proiect (exceptând fişierele sursă ale programelor IEC) sunt memorate într-o bază de date MS-Access.

Un concept folosit este "Program Organization Unit" (POU) − un set de instrucţiuni scrise într-unul din limbajele de mai sus, sau în limbajul IEC 61499. Unităţile de program pot fi programe, funcţii, sau funcţii bloc. Programele sunt executate pe sistemul de calcul ţintă, respectând ierarhia de programe din resurse. Orice program poate apela o funcţie. Funcţiile pot fi programate doar în limbajele ST, LD, sau FBD. Intr-o funcţie pot fi declarate doar variabile locale (posibil structuri sau tablouri), care nu pot fi însă instanţe ale unei funcţii bloc. La fiecare execuţie a unei funcţii, variabilele sale locale sunt repuse la valorile lor iniţiale (zero, dacă nu sunt date valori într-un dicţionar). Un program sau o funcţie bloc, dar nu o funcţie, poate apela o altă funcţie bloc. Funcţiile bloc sunt scrise în limbajele SFC, ST, LD, sau FBD, dar şi în limbajul IEC 61499. Funcţiile bloc SFC pot avea descendenţi de aceeaşi natură. Ordinea în care sunt definite funcţiile sau funcţiile bloc în secţiunea lor este arbitrară. Interfaţa grafică permite crearea de unităţi de program din meniul principal sau meniul contextual accesând componenta respectivă dintr-o resursă. După creare, o unitate de program poate fi plasată grafic într-o nouă poziţie din aceeaşi secţiune, sau dintr-o altă secţiune sau resursă.

Imaginea arhitecturii hardware redă grafic configuraţiile unei proiect şi legăturile de reţea dintre ele, permiţând administrarea diverselor aspecte:

- crearea de configuraţii sau reţele;

- ataşarea „ţintelor” (calculatoare destinaţii) la configuraţii;
- inserarea resurselor;
- deplasarea resurselor între configuraţii;
- conectarea configuraţiilor şi reţelelor;
- definirea proprietăţilor de conectare; etc.

Reţelele oferă mijloacele de comunicaţie între configuraţii. Calculatorul ţintă ataşat unei configuraţii trebuie să suporte reţeaua la care este conectată configuraţia. Nu se limitează numărul de reţele ale unui proiect. Proprietăţile reţelei sunt definite la creare. Fiecare configuraţie trebuie conectată la una sau mai multe reţele şi reciproc. Se pot defini una sau mai multe ţinte alternative, care au aceiaşi parametrii de reţea ca şi ţinta principală.

Dicţionarul este un instrument de editare folosind arbori şi grile pentru declararea variabilelor, funcţiilor şi a parametrilor funcţiilor bloc, cât şi cuvintele definite ale unui proiect. Există arbori de variabile (globale şi locale), parametri, tipuri (structuri sau tablouri) şi cuvinte definite.

Interconexiunile de intrare/ieşire (I/O) permit stabilirea legăturilor dintre variabilele definite într-un proiect şi canalele dispozitivelor I/O de pe sistemul ţintă. Este necesară ataşarea ţintei la configuraţia curentă şi selectarea unei resurse fie în arhitectura conexiunilor, fie în arhitectura hardware. Se poate defini transformarea de la canalele logice şi fizice.

Bibliotecile sunt proiecte speciale formate din configuraţii şi resurse în care se definesc funcţii şi funcţii bloc reutilizabile în alte proiecte. Bibliotecile permit modularizarea proiectelor şi izolarea funcţiilor (bloc) aşa încât acestea să poată fi testate şi validate separat. La definirea dependenţelor unui proiect se specifică bibliotecile utilizate. Bibliotecile pot conţine unităţi de program, definiţii de variabile globale şi alte elemente utilizate pentru testarea funcţiilor şi funcţiilor bloc. Funcţiile (bloc) dintr-o bibliotecă sunt compilate de programul apelant, pentru a utiliza opţiunile de compilare definite în proiect. O bibliotecă nu poate folosi funcţii dintr-o altă bibliotecă, ceea ce constituie o limitare.

Există două moduri de lucru: simulare şi execuţie online. In modul de simulare, intrările şi ieşirile nu sunt administrate pe maşina ţintă, iar fiecare resursă este executată pe o maşină virtuală de pe calculatorul care rulează bancul de lucru. In modul online, fiecare resursă este executată pe o maşină virtuală de pe platforma reală. Codul fiecărei resurse trebuie descărcat în prealabil pe acea platformă. O resursă poate fi modificată în timpul execuţiei, dar acest lucru nu este în general recomandabil. O modificare online constă în înlocuirea unei sau mai multor secvenţe de cod (adică, un set complet de instrucţiuni ST, IL, LD, FBD 61131, sau FBD 61499, ori un pas/o acţiune sau tranziţie/test într-un program SFC sau FC), fără a întrerupe ciclul de execuţie al unui PLC.

Variabilele logice (booleene) sunt afişate utilizând culori; culorile implicite sunt roşu pentru valoarea adevărat şi albastru pentru fals.

Un proiect de tip IEC 61499 conține funcții bloc distribuite pe diferite resurse. Este necesară setarea dependenței față de librăria standardului IEC 61499 ce este instalată o dată cu ISaGRAF. O aplicație poate conține una sau mai multe bucle de control, unde datele de intrare sunt încărcate pe un dispozitiv, procesul de control se execută pe alt dispozitiv, iar afișarea rezultatelor de ieșire se face pe un alt dispozitiv. Aceste bucle de

control, care colaborează între ele, partajează date într-un mod explicit, detaliat în standardul IEC 61499. Orice program poate fi o aplicație distribuită. Figura 4 arată distribuirea unor aplicații pe diverse resurse. Acesta este modelul de sistem afișat de către mediul de dezvoltare ISaGRAF. Funcțiile bloc sunt afișate în culoarea galbenă. Fiecare element al unei aplicații este conectat către altele prin rețeaua de comunicații. O dată cu dezvoltarea unei aplicații, ISaGRAF face automat legăturile între elementele acesteia.

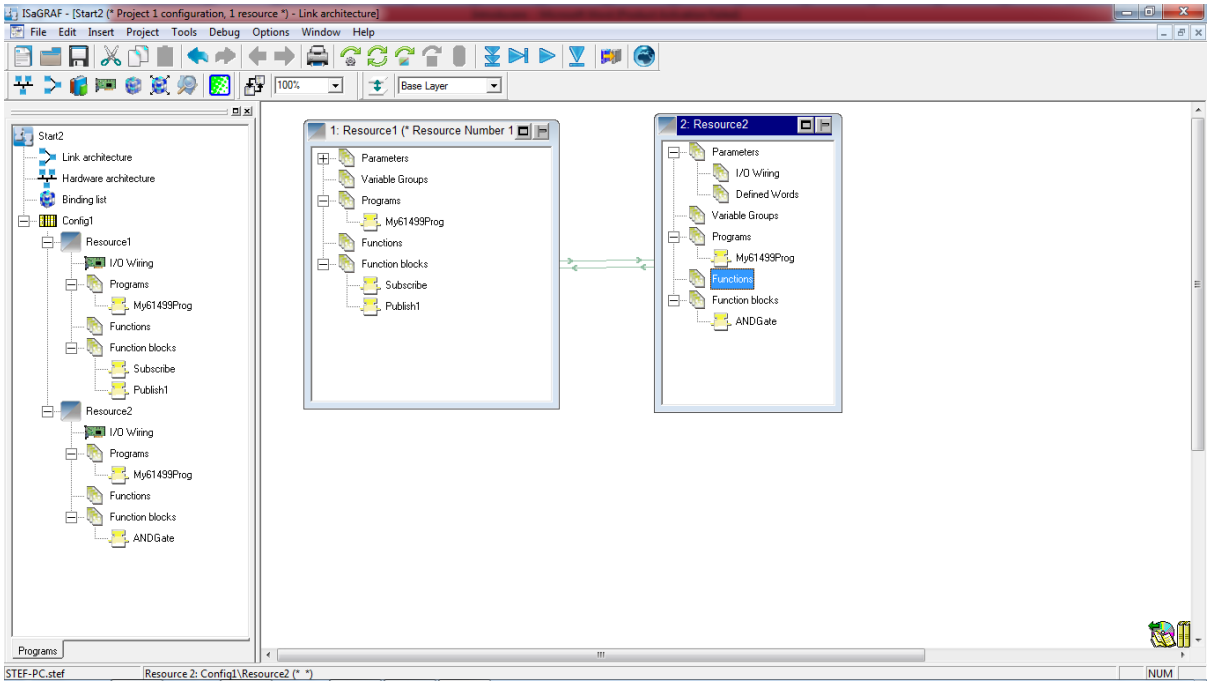



Fig. 4. Afișarea legăturilor între elementele unei aplicații

Un model de resurse prezintă părți incluse în resursa de control. Figura 5 arată cum sunt afișate aceste blocuri în ISaGRAF. Multe dintre funcțiile bloc sunt interconectate cu o interfață de tip I/O și fac parte dintr-o resursă. O resursă este considerată ca fiind o unitate funcțională încorporabilă într-un dispozitiv. Funcțiile unei resurse sunt de a accepta valorile de intrare, de a procesa datele și de a furniza valorile de ieșire. Un dispozitiv reprezintă un sistem hardware capabil să execute bucle de control programate în una sau mai multe resurse.

O funcție bloc reprezintă o unitate funcțională a unui algoritm. Algoritmii incluși într-o funcție bloc nu sunt afișați în funcția bloc și sunt programați cu ajutorul mașinii de stări ECC (Execution Control Chart), Fig. 6. Evenimentele de intrare și ieșire sunt folosite pentru sincronizarea funcțiilor bloc într-o aplicație, prin planificarea algoritmilor dintr-o funcție bloc. Datele de intrare și ieșire reprezintă interfața cu partea externă a funcției bloc, deoarece datele interne sunt ascunse. Datele unei funcții bloc pot fi parte din algoritmi sau pot fi stări de informație. Funcțiile bloc sunt create prin programarea algoritmilor acestora în fereastra ECC. În această fereastră este definit comportamentul unei funcții bloc la primirea unor date. Algoritmii programați pot funcționa pe valori interne, valori de intrare și valori de ieșire. Fiecare funcție bloc poate opera pe orice resursă. În timpul rulării unei aplicații se observă interdependențele fiecărei funcții bloc și schimbul de date între acestea (Fig. 7).

ISaGRAF este unul dintre puținele sisteme de dezvoltare a programelor pentru controllere ce includ standardul IEC 61499. Totuși acest sistem nu a mai fost actualizat din

anul 2012 și permite o funcționabilitate scăzută pe sistemele de operare mai noi decât Windows Vista. ISaGRAF se poate descărca în varianta demo de pe platforma www.ISaGRAF.com.

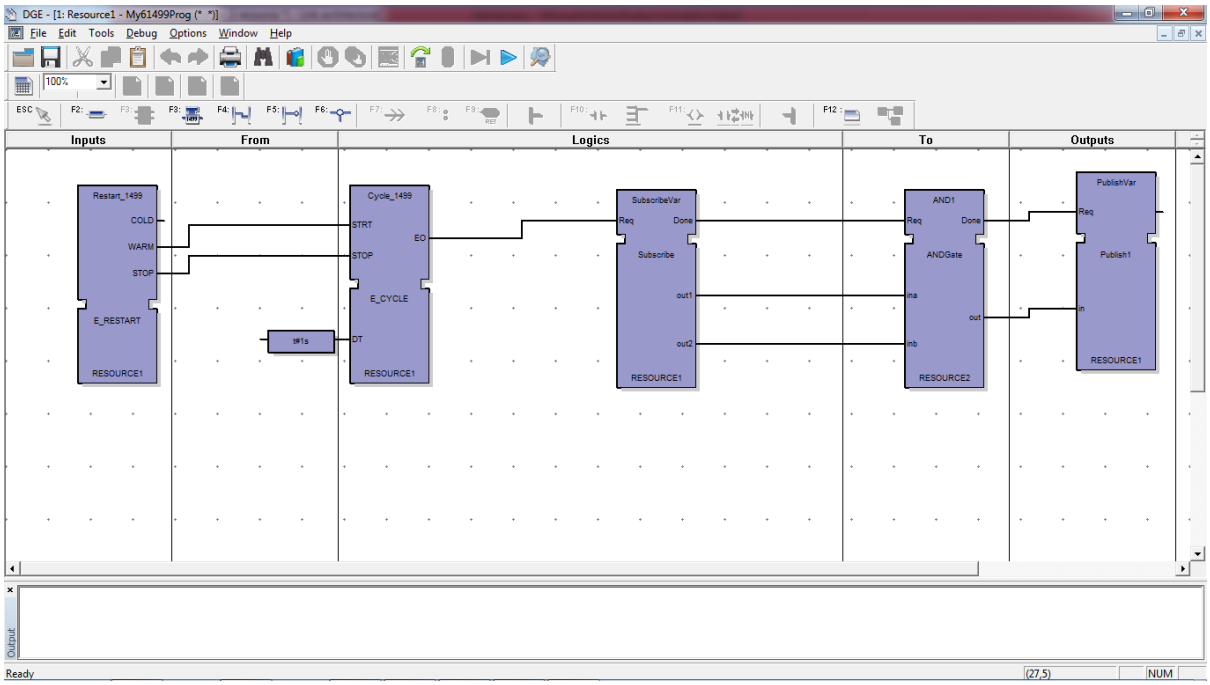



Fig. 5. Afișarea funcțiilor bloc în ISaGRAF

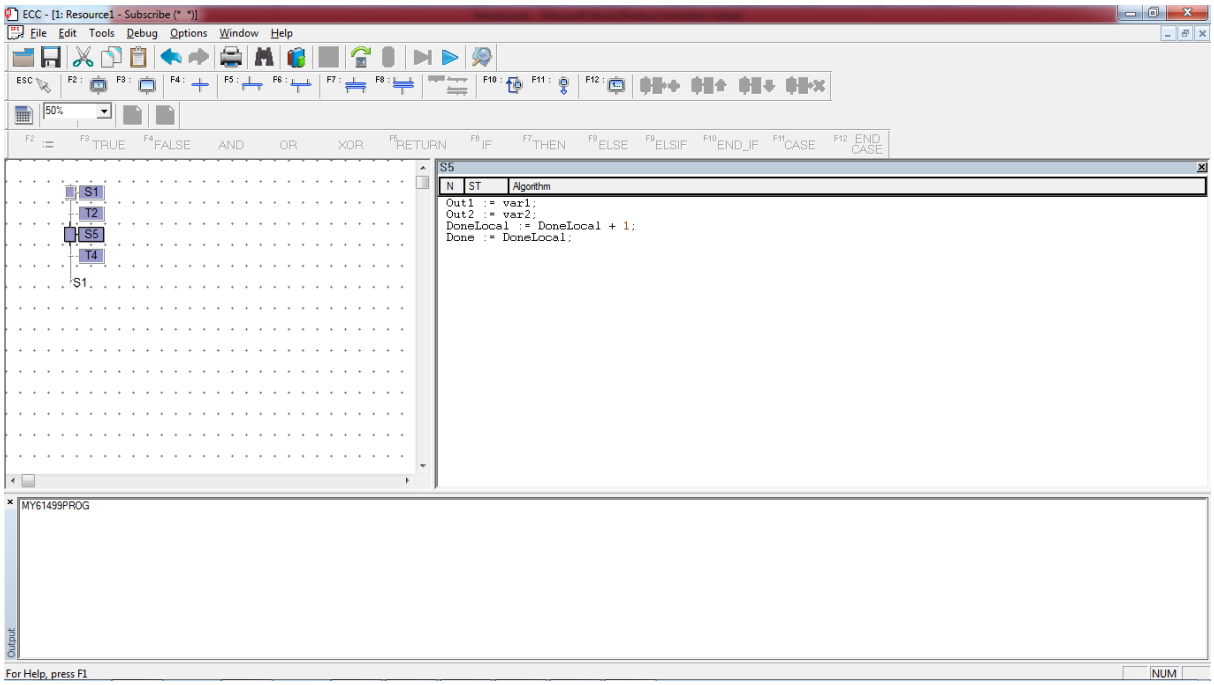



Fig. 6. Programarea algoritmilor unei funcții bloc

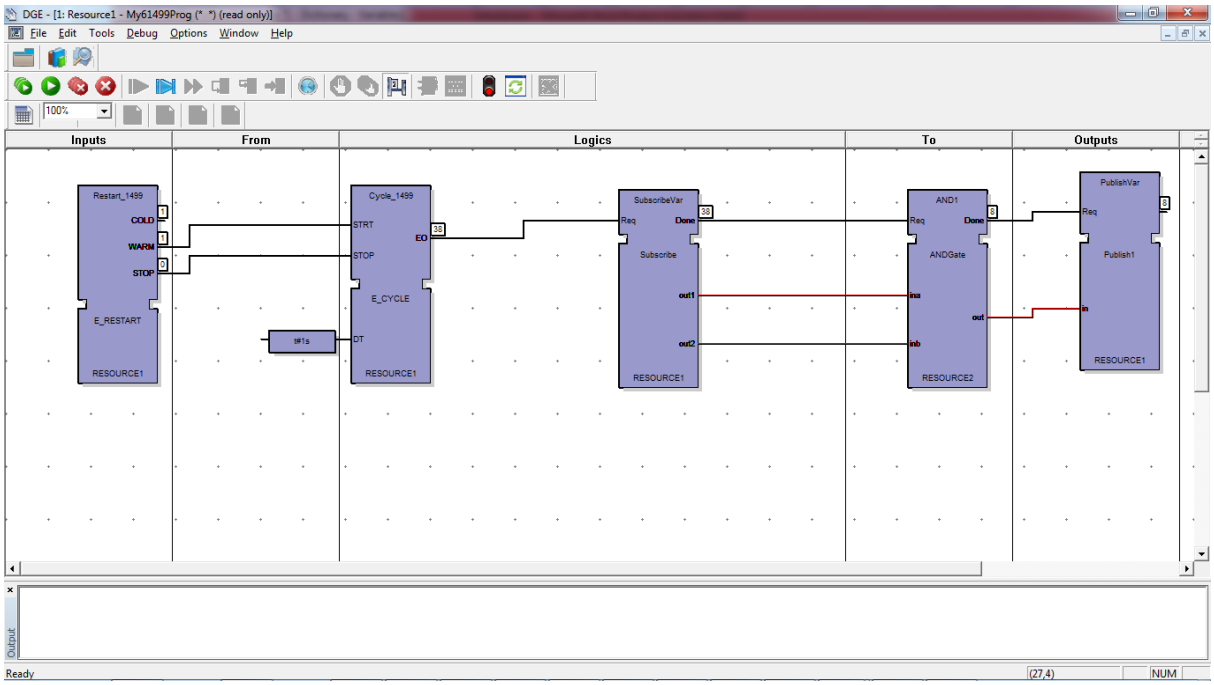



Fig. 7. Se observă creșterea valorii de ieșire a funcției bloc AND1 când aceasta primește valori de intrare pozitive

# 3. Fundamentele teoretice şi posibilităţile de implementare

## 3.1. Algoritmi şi modelare

Automatica este o ştiinţă formalizată bazată pe diferite domenii ale matematicii. Datorită limitărilor de spaţiu, nu este posibilă includerea în acest capitol decât a unei scurte treceri în revistă a problematicii algoritmice relevante pentru proiectul CALCULOS. O categorie de algoritmi o constituie cei de identificare experimentală [60-66 şi referinţele citate]. O altă categorie o formează algoritmii de optimizare, în special, cei de optimizare liniar-pătratică şi tehnicile numerice asociate [67-72]. Evoluţiile recente includ exploatarea structurii problemelor de calcul optimale (implicând, de pildă, matrice Hamiltoniene sau simplectice, sau fascicole de matrice anti-Hamiltoniene/Hamiltoniene) [73]. De mare interes sunt sistemele tolerante la defecte şi sistemele optimale reconfigurabile [74-88], pentru care sunt adecvaţi algoritmi iterativi, de pildă, de tip Newton [89,90]. Multe detalii se găsesc în referinţele citate. In continuare, vom prezenta succint, dar într-un context mai general, problema anomaliilor în funcţionarea unui proces [91].

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.

*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 [91]. Instrumentul principal folsosit este testarea ipotezelor statistice. In continuare, ne vom referi la abordarea de implementare a soluţiilor folosind sisteme de calcul moderne, de mare performanţă, bazate pe Cloud Computing.

## 3.2. Calculul în "cloud" (cloud computing)

Calculul în "cloud" (cloud computing) este calculul bazat pe Internet în care grupuri mari de calculatoare (servere) sunt interconectate pentru a permite stocarea centralizată a datelor şi accesul direct (online) la serviciile sau resursele de calcul (http://en.wikipedia.org/wiki/Cloud_computing). Platformele cloud pot fi clasificate ca publice, private, sau hibride. Resursele din cloud sunt, de regulă, partajate de mai mulţi utilizatori şi realocate dinamic la cerere. Oracle Cloud, lansat în iunie 2012, este considerat a fi primul cloud furnizând utilizatorilor accesul la un set integrat de soluţii ale tehnologiei informaţiei, incluzând nivelurile de aplicaţie, platformă, şi infrastructură. Principala tehnologie inovatoare pentru calculul în cloud este *virtualizarea*, care permite divizarea unui dispozitiv de calcul fizic în unul sau mai multe dispozitive „virtuale", putând fiecare să fie uşor utilizat şi administrat pentru a efectua activităţi de calcul. Devine posibilă optimizarea utilizării resurselor de calcul. Toate resursele cloud sunt oferite ca servicii şi se folosesc standarde larg acceptate şi cele mai bune practici dobândite în domeniul arhitecturilor bazate pe servicii (SOA – Service Oriented Architecture), pentru a permite accesul global şi uşor la capabilităţile de

calcul în cloud. In unele oferte de platformă, ca Microsoft Azure şi Google App Engine, resursele de calcul şi memorie se adaptează automat la cerinţele aplicaţiei, aşa încât utilizatorul nu trebuie să le aloce manual. Google App Engine a fost, de asemenea, propus ca arhitectură ţintind la facilitarea aplicaţiilor în timp-real. Un cloud hibrid este o compunere a două sau mai multe sisteme cloud (private, comunitare sau publice) care rămân entităţi distincte, dar sunt conectate, oferind beneficiile modelelor cu implementări multiple. Un cloud hibrid poate utiliza un model de implementare a aplicaţiilor numit "explozia cloud", care permite crearea unei infrastructuri informatice care suportă încărcările de calcul medii, dar utilizează resursele unui cloud public sau privat în timpul vârfurilor de încărcare.

CALCULOS va fi construit pe o platformă sub forma unui server în cloud care conţine o interfaţă prietenoasă API şi foloseşte obiecte avansate de calcul şi comunicaţie. Această platformă va oferi suport pentru dezvoltarea, într-un timp mai scurt, a unor algoritmi cu eficienţă superioară, pe baza funcţiilor disponibile, caracterizate prin faptul că sunt reutilizabile şi deschise; de asemenea, va permite utilizatorului să ruleze algoritmi de complexitate sporită, pentru a fi folosiţi pe controllere cu resurse limitate, implementând obiecte de comunicaţie şi servicii cloud pentru executarea funcţiilor. In acord cu tendinţa pe plan mondial, se doreşte implementarea unor algoritmi sub formă de funcţii bloc, conforme cu standardele actuale, de pildă IEC 61499. Baza pentru obţinerea unei arhitecturi pentru algoritmi complecşi ce rulează sub standardul de funcţii bloc este inspirat din [18-20]; de asemenea, interesul partenerilor industriali pentru acest tip de servicii şi felul în care influenţează activitatea industrială sunt evidenţiate în [15, 17]. Se propune ca platforma să fie construită pe un cloud de tip hibrid, pentru a permite accesul nu doar în cadrul grupului, ci şi în afara grupului. Pentru a reduce timpul de dezvoltare şi a îmbunătăţi calitatea proiectării, se doreşte implementarea unui mediu integrat ce va oferi suport pentru o eficienţă mai bună în fazele de modelare, prin adăugarea accesului la funcţii avansate de control, la reacţiile altor utilizatori ai platformei după implementarea algoritmilor în diferitele instalaţii şi industrii, şi la instrumentele de dezvoltare.

Pentru anii următori, se prevede o creştere semnificativă a capacităţii de calcul în cloud [92]. Există mai multe tipuri de servicii disponibile pentru utilizare în cloud: Network as a service, Storage as a service, Data as a service, Database as a service, Test environment as a service, Desktop virtualization, API as a service, Backend as a service, servicii Comune etc., dar cele mai frecvent folosite sunt cunoscute ca servicii SPI:

- Software as a service (SaaS)
- Platform as a service (PaaS)
- Infrastructure as a service (IaaS)

IaaS reprezintă modelul fundamental în care o companie îşi pune la dispoziţie partea de hardware ce asigură suport pentru stocare, comunicaţie şi capacitatea de calcul. Cele mai renumite companii de hardware din lume deja îşi pun la dispoziţie cele mai noi modele de echipamente pentru a-şi integra produsele în arhitecturile cloud. PaaS este permis de către modelul IaaS şi oferă o arhitectură de tip cloud ce asigură şi sistemele de operare şi platformele software pe care rulează aplicaţiile utilizatorilor. Modelul SaaS oferă un nivel mai ridicat de abstractizare pe PaaS şi se bazează pe furnizarea unor servicii software specifice. Unul dintre avantajele procesării în cloud este posibilitatea de a înlocui cheltuielile mari cu infrastructura, cu costuri reduse şi variabile ce se dimensionează după tipul afacerii. Un alt avantaj important al serviciilor bazate pe cloud îl reprezintă mobilitatea lor, limitată numai de reţeaua de comunicaţie. S-au format sau s-au dezvoltat

câteva companii care profită de pe urma acestor oportunităţi, concurând la implementarea arhitecturilor de tip cloud şi a serviciilor SPI: Google, Amazon, VMWare, Citrix Systems, Microsoft, Rackspace, Salesforce, Verizon. Proiectul va include platforma colaborativă pentru a utiliza procesarea cloud a algoritmilor.

## 3.3. Soluţii comerciale şi open-source pentru utilizarea serviciilor cloud de tip Platform-as-a-Service (PaaS)

Tehnologia „cloud computing" devine din ce în ce mai populară în industrie, iar companiile utilizează soluţii precum IaaS, PaaS, precum şi SaaS [1]. Soluţiile cloud SaaS sunt utilizate în mod obişnuit, deoarece acestea oferă servicii bine stabilite, cum ar fi e-mail, mesagerie instant, calendar comun şi gestiunea documentelor (oferite, de exemplu, de Google Apps şi Microsoft Office 365). Deşi modelul SaaS este excelent pentru astfel de servicii, totuşi acesta nu este la fel de flexibil şi uşor de personalizat. Modelul IaaS oferă accesul de la distanţă (prin Internet) la infrastructuri de calcul, de obicei sub forma de sisteme virtuale (maşini virtuale sau VM-uri). Utilizatorii IaaS trebuie să administreze întreaga stivă software pentru a rula propriile aplicaţii. Astfel, modelul IaaS este potrivit pentru acele proiecte care necesită o configurare specială sau pentru a construi infrastructuri cloud de tip PaaS utilizând suportul IaaS. Soluţiile IaaS necesită, în plus, efectuarea unor operaţiuni regulate de întreţinere, cum ar fi aplicarea actualizărilor de securitate pentru sistemul de operare de bază şi mediul de execuţie al aplicaţiilor. Aplicaţia utilizator poate deveni vulnerabilă atunci când această întreţinere regulată nu este realizată corect.

Modelul PaaS reprezintă un compromis avantajos între nivelurile IaaS şi SaaS. Acesta oferă platforma de execuţie şi toate serviciile necesare (de exemplu, o bază de date sau server de aplicaţii) pentru aplicaţiile existente, menţinând în acelaşi timp sistemul de operare şi mediul de execuţie. Soluţiile PaaS sprijină, de asemenea, capacitatea de scalare verticală prin creşterea sau scăderea resurselor (RAM, CPU, etc), care sunt acordate către o singură instanţă. Scalarea verticală poate fi asigurată şi de nivelul IaaS de bază. Mai multe platforme PaaS comerciale oferă capacităţi de scalare orizontale prin adăugarea sau eliminarea de instanţe suplimentare pentru o aplicaţie distribuită.

Serviciile de tip IaaS şi PaaS sunt luate în considerare mai rar, în timp ce serviciile de găzduire clasice sunt încă la modă, beneficiind de popularitatea lor crescută din rândul utilizatorilor. Majoritatea acestor utilizatori nu necesită servicii la nivelul IaaS pentru că pur şi simplu doresc să ruleze anumite aplicaţii, însă serviciile de tip PaaS devin din ce în mai căutate deoarece oferă un raport mai avantajos între flexibilitatea şi uşurinţa în utilizare a aplicaţiilor.

Odată cu expansiunea tehnologiilor cloud computing s-a diversificat în ultimii ani şi numărul furnizorilor de servicii şi soluţii aferente. În continuare, vom prezenta cele mai importante soluţii „open-source" pentru servicii PaaS, rezumând caracteristicile acestora precum şi unele probleme de portare ale aplicaţiilor, restricţii, politici de stabilire a preţurilor şi procesul de instalare locală.

### Google App Engine

Una dintre cele mai vechi platforme cloud PaaS este Google App Engine (GAE). Lansată în 2008, GAE oferă un mediu de rulare pentru aplicaţii web în centrele de date administrate

de Google. GAE acceptă în prezent limbajele Java, Python, PHP şi Go. Java şi Python sunt acceptate oficial (cu unele restricţii) în timp ce PHP este suportat nativ, iar Go este experimental. GAE rulează în prezent aplicaţii Java folosind container-ul web Jetty. Sunt prevăzute mai multe metode de stocare a datelor. Cloud Datastore este un depozit de date NoSQL complet scalabil bazat pe tehnologia Bigtable. Google Cloud Storage este un sistem de stocare on-line de obiecte, de obicei fişiere, de tip RESTful. Aceste obiecte sunt organizate în containere şi permisiunile pot fi definite în listele de control al accesului (ACL-uri). Cloud SQL este o bază de date relaţională similară MySQL cu suport de replicare a datelor.

Există restricţii semnificative impuse de mediul de execuţie GAE, care sunt rezonabile în mediul cloud. Cu toate acestea, ele pot complica portarea aplicaţiilor existente la GAE. În primul rând, este disponibil numai un subset din clasele standard JRE. Aplicaţiile care folosesc alte clase decât cele oferite de GAE trebuie să fie rescrise şi recompilate. Aplicaţiile nu pot utiliza sistemul de fişiere local şi trebuie să fie rescrise pentru a utiliza Google Cloud Storage, Datastore sau o altă tehnologie specifică GAE.

Utilizatorii care rulează aplicaţii pe platforma GAE trebuie să plătească în funcţie de resursele consumate. Spre exemplu, cea mai mică instanţă are 128MB RAM şi un spaţiu de stocare de 5GB utilizând Google Cloud Storage. Google are o listă de preţuri cu o granulaţie foarte fină în care resursele specifice sunt facturate per gigabyte, pe oră, pe gigabytes pe lună, etc., în funcţie de natura resurselor (trafic în reţea, performanţă maşină virtuală, stocare date, etc). De asemenea, este posibil să se ruleze în mod gratuit aplicaţii atunci când nu se depăşeşte o cotă maximă de resurse care sunt alocate gratuit.

Platforma open-source AppScale, creată la Universitatea din California, permite dezvoltatorilor să implementeze şi să ruleze aplicaţii construite pentru GAE oriunde în afara ecosistemului Google, de exemplu, pe propriile servere (virtuale), pe maşini virtuale oferite de furnizori terţi, cum ar fi Amazon sau pe maşini virtuale gestionate de tehnologii deschise, cum ar fi OpenStack sau Eucalyptus. AppScale foloseşte unele părţi ale GAE disponibile în Google App Engine Software Development Kit (SDK). Acest SDK este oferit de Google şi permite dezvoltatorilor să testeze şi să ruleze aplicaţiile GAE la nivel local, fără a fi necesară implementarea lor efectivă în cloud. Pe lângă SDK-ul GAE, AppScale utilizează şi alte proiecte open-source existente, de exemplu Cassandra (bază de date NoSQL), MySQL (bază de date relaţionale), MongoDB, HBase şi altele.

### OpenShift

OpenShift este un sistem PaaS dezvoltat de Red Hat, lansat iniţial în 2011, sub forma a trei ediţii: OpenShift Origin este varianta open-source şi care este compatibilă cu celelalte ediţii, OpenShift Online este serviciul cloud public comercial şi OpenShift Enterprise poate fi implementat în centrul de date al companiei sau într-un un cloud privat. OpenShift acceptă mai multe limbaje de programare, baze de date, servicii şi aplicaţii, care sunt capabile să ruleze pe sistemul Red Hat Linux. Suportul pentru o anumită tehnologie este întotdeauna încapsulat sub forma unei componente denumite cartuş (cartridge). Mai multe cartuşe sunt disponibile pentru Java Runtime (JBoss, Tomcat), mediul de execuţie PHP (Apache cu mod_php), baze de date relaţionale (MySQL, PostgreSQL), baze de date NoSQL (MongoDB), servicii (Cron, Jenkins), etc. Aplicaţiile existente pot fi, de asemenea, ambalate în cartuşe pregătite să permită unor non-dezvoltatori să ruleze aplicaţiile favorite de genul WordPress în cadrul platformei OpenShift (sunt necesari doar câţiva paşi de configurare). Se pot crea, de asemenea, propriile cartuşe personalizate, deşi multe cartuşe

suplimentare au fost deja implementate de către comunitatea de utilizatori OpenShift. Spre deosebire de platformele care sunt strâns cuplate cu anumite tehnologii (cum ar fi de exemplu Google App Engine), OpenShift este deschis către orice tehnologie.

Pentru a satisface aceste facilităţi de permisivitate este strict necesară aplicarea unei politici de securitate foarte puternice. Prin urmare, cartuşele necesare sunt implementate în aşa-numita componentă Gear. Aceasta este un mediu izolat, construit pe baza mai multor directoare, ce utilizează tehnologia cgroups Linux (de punere în aplicare a limitării utilizării resurselor) şi politicile SELinux (consolidarea securităţii şi izolare). Deoarece componenta OpenShift Gear este de fapt un sistem GNU/Linux containerizat, restricţiile impuse sunt minimale. Pe de altă parte, din cauza acestei arhitecturi simple, nu există soluţii „avansate" pentru scalarea orizontală a sistemului de stocare de fişiere şi/sau a bazelor de date relaţionale. Chiar dacă este permis să scrie în sistemul local de fişiere, se garantează persistenţa modificărilor în decursul ciclului de viaţă al unei aplicaţii doar pentru directorul OPENSHIFT_DATA_DIR, fapt ce poate fi considerat drept o limitare. Astfel, toate datele persistente ale unei aplicaţii ar trebui să fie depozitate în acest director, dar acesta nu este replicat în cadrul celorlaltor containere Gear atunci când se solicită scalarea aplicaţiei. Un nou container va avea directorul de date gol.

Codul sursă al aplicaţiei este întotdeauna obţinut dintr-un repository al sistemului de gestiune al codului sursă Git. Când se face portarea unei aplicaţii non-scalabile pentru platforma OpenShift, aceasta trebuie să fie modificată pentru a stoca toate fişierele de date în directorul de date. Se poate implementa, de asemenea, ca baza de date necesară unei aplicaţii să se afle în acelaşi container. Însă acest lucru nu reprezintă un scenariu pentru o implementare scalabilă. Spre deosebire de Google App Engine, OpenShift nu are un suport direct pentru scalarea/replicarea MySQL. Aplicaţiile scalabile ar trebui să utilizeze o bază de date comună şi care să fie implementată într-un alt container. Iar containerul dedicat pentru această bază de date ar trebui să fie suficient de puternic pentru a evita o limitare a performanţei.

OpenShift utilizează containere în trei dimensiuni şi care au preţuri diferite în mod corespunzător. În prezent există trei planuri de utilizare a platformei publice OpenShift Online. Planul „gratuit" permite dezvoltatorilor să ruleze aplicaţiile utilizând 3 containere mici (512MB RAM şi 1 GB de stocare fiecare) fără niciun cost. Planul „bronz" este similar cu cel gratuit, dar utilizatorul poate comanda containere suplimentare de depozitare (mici, medii sau mari) şi spaţiu de stocare suplimentar, facturat pe oră sau per gigabyte pe lună. Planul „argint" adaugă spaţiu de stocare suplimentar pentru o taxă lunară, permite utilizarea a mai mult de 16 containere şi oferă suport.

OpenShift Origin este versiunea open source a produsului OpenShift Enterprise şi a serviciului OpenShift Online, urmând tradiţia Red Hat de a oferi varianta de dezvoltare a produselor sale comerciale către comunitatea open-source. Relaţia dintre OpenShift Origin şi OpenShift Enterprise este foarte similară cu cea dintre Red Hat Fedora şi Red Hat Enterprise Linux.

Pentru instalare şi testare rapidă, Red Hat oferă imagini configurate pentru VirtualBox şi KVM, ce permit unui utilizator să folosească OpenShift în cadrul unei maşini virtuale. Există şi script-uri shell pentru instalarea generică în cadrul sistemelor Red Hat, iar pentru o instalare complet automată se poate utiliza instrumentul Puppet.

## Cloud Foundry

Cloud Foundry este o platformă cloud open source dezvoltată de VMware şi lansată în anul 2011. La început, Cloud Foundry a oferit suport pentru limbajele Java, Scala, Ruby şi Node.js. În ultima versiune a platformei a fost introdus un nou element care oferă suport pentru execuţia oricărei aplicaţii. Această caracteristică se bazează pe tehnologia Heroku Buildpacks care automatizează împachetarea unei aplicaţii împreună cu mediul său de execuţie. Cloud Foundry se bazează pe două tipuri de servicii: servicii utilizator şi servicii de administrare. Serviciile pentru utilizator permit unui dezvoltator de aplicaţii să furnizeze un serviciu extern existent (de exemplu, o bază de date Oracle) către o aplicaţie. Serviciile de administrare sunt servicii gestionate şi implementate de platforma Cloud Foundry. Unele baze de date relaţionale (MySQL, PostgreSQL), baze de date NoSQL (MongoDB) sau alte servicii (Memcached) sunt suportate implicit. Serviciul MySQL cu disponibilitate ridicată este oferit de un serviciu terţ numit ClearDB.

Cloud Foundry este o platformă deschisă care permite execuţia aplicaţiilor fără constrângeri speciale, ceea ce face ca politicile de securitate puse în aplicare de către platformă să fie foarte stricte. Cloud Foundry foloseşte o componentă specială denumită Warden, respectiv un nivel de abstractizare care gestionează resursele disponibile în containere izolate şi care limitează utilizarea resurselor disponibile aplicaţiilor care sunt executate în cadrul platformei. Deşi sistemul Warden este similar cu tehnologia Linux Containers (LXC), acesta este proiectat special pentru a fi multi-platformă. Cu toate acestea, implementarea de referinţă suportă numai sistemele GNU/Linux care utilizează tehnologia cgroups din nucleul Linux.

## Interoperabilitatea infrastructurilor cloud

Tehnologia cloud computing reprezintă o nouă paradigmă de calcul prin care se oferă posibilitatea de a accesa la cerere anumite resurse (spre exemplu CPU, spaţiu de stocare, reţea, baze de date, aplicaţii) şi pentru care se plăteşte doar costul utilizării efective. Prin provizionarea rapidă şi eliberarea resurselor cu un minim de efort din partea utilizatorului şi de interacţiune cu furnizorul de servicii cloud, această tehnologie facilitează implementarea unor soluţii de calcul scalabile şi eficiente. Modelul cloud computing are următoarele caracteristici esenţiale:

- este disponibil la cerere;
- se bazează pe auto-servirea de către utilizator;
- uşor accesibil prin intermediul Internet-ului;
- partajarea resurselor de calcul;
- elasticitate şi contabilizarea utilizării serviciilor.

Tehnologia cloud este extrem de utilă în diverse aplicaţii, inclusiv cele care sunt mari consumatoare de resurse de calcul sau de spaţiu de stocare, ori cele care necesită o lărgime de bandă foarte mare [94]. Furnizorii de servicii cloud au proliferat rapid în ultimii 6-7 ani, cei mai importanţi fiind Amazon, Google, Microsoft şi Salesforce. Fiecare dintre aceştia îşi promovează propria infrastructură cloud, precum şi standarde şi formate incompatibile pentru accesarea resurselor cloud.

Din punctul de vedere al utilizatorilor, clienţii cloud sunt de obicei reticenţi în a fi legaţi de serviciile cloud oferite de un singur furnizor, care nu ar fi în măsură să satisfacă toate cerinţele utilizatorilor, cum ar fi distribuţia geografică a centrelor de date, oferirea de Service Level Agreements (SLAs), etc., existând riscul potenţial de creştere exagerată a

costurilor ca urmare a dependenţei de un singur furnizor. Utilizatorii cloud doresc infrastructuri cloud interoperabile, în care să poată avea un control deplin asupra modului de instalare a aplicaţiilor şi, de asemenea, să poată să migreze cu uşurinţă atunci când este nevoie, fără investiţii de dezvoltare suplimentare.

Din punctul de vedere al furnizorilor de servicii cloud, această incompatibilitate între furnizorii de soluţii cloud poate proteja temporar interesul fiecărui furnizor, însă nu şi pe termen lung, în cazul în care piaţa de servicii cloud se maturizează. În acest sens, au apărut iniţiative pentru stabilirea unor standarde pentru federalizarea infrastructurilor cloud deţinute de diferiţi furnizori, în special susţinute de către furnizorii de servicii cloud relativ mici (de exemplu, Rackspace, GoGrid), dar şi de către cei nou intraţi pe piaţă (de exemplu, Red Hat, Dell, Oracle, etc.).

Interoperabilitatea infrastructurilor cloud reprezintă un trend promiţător al acestei tehnologii, deoarece promovează mai bine îndeplinirea scopului final al paradigmei cloud computing, şi anume acela de furnizare la scară globală, „nelimitată”, cu acces prin interfeţe unificate, a resurselor de calcul. Importanţa interoperabilităţii cloud a fost evidenţiată atât de către industrie, cât şi de către mediul academic. Industria încearcă să abordeze problemele de interoperabilitate cloud prin standardizare. O altă soluţie pe termen scurt, promovată de unii cercetători atât în industrie cât şi din mediul academic, a fost dezvoltarea de soluţii tehnologice care să permită interoperarea între anumite infrastructurii cloud, eficiente atât din perspectiva furnizorului de servicii cloud, cât şi a utilizatorului.

Interoperabilitatea se referă, în general, la capacitatea unor diferite sisteme eterogene de a funcţiona şi de a interacţiona unele cu altele. Potrivit Comisiei Europene (CE), interoperabilitatea este definită ca fiind capacitatea sistemelor şi proceselor din domeniul Tehnologiei Informaţiei şi Comunicaţiilor (TIC) de a face schimb de date şi de a permite distribuirea de informaţii şi cunoştinţe. În cadrul tehnologiei cloud, interoperabilitatea poate fi definită ca fiind capacitatea de a utiliza aplicaţiile, SLA-urile, modurile de autentificare şi autorizare între diferite infrastructuri cloud astfel încât acestea să poată coopera sau interopera. Interoperabilitatea, compatibilitatea şi portabilitatea în cloud sunt noţiuni strâns legate şi pot fi adesea confundate. O clarificare privind asemănările şi diferenţele dintre aceşti termeni a fost realizată de Cohen în [95], astfel:

- interoperabilitatea cloud reprezintă abilitatea mai multor furnizori de servicii cloud de a lucra împreună;
- atât compatibilitatea cloud, cât şi portabilitatea cloud, răspund la întrebarea „cum?”: compatibilitatea cloud se referă la faptul că aplicaţiile şi datele pot fi utilizate în acelaşi mod, indiferent de furnizorul de servicii cloud, în timp ce portabilitatea cloud reprezintă capacitatea de a migra şi refolosi datele şi aplicaţiile indiferent de alegerea furnizorului de servicii cloud, a sistemului de operare, ori a formatului de stocare a datelor sau API-urilor.

O înţelegere cât mai clară şi clasificarea cerinţelor care asigură interoperabilitatea serviciilor cloud reprezintă primul pas spre standardizarea platformelor cloud computing, a API-urilor şi serviciilor cloud.

Pe baza celor trei modele de servicii cloud adoptate pe scară largă de către comunitatea cloud, o abordare semantică în delimitarea interoperabilităţii cloud computing a fost prezentată de către Sheth şi Ranabahu [96], astfel:

- Infrastructura ca Serviciu (IaaS). Capacitatea furnizată consumatorului cloud este

de resurse computaţionale, stocare date, crearea de reţele, şi alte resurse de calcul fundamentale, consumatorul fiind capabil de a implementa şi de a rula orice software, precum sisteme de operare şi aplicaţii. Exemple de servicii IaaS sunt Amazon EC2 şi Google Compute Engine.

- Platforma ca Serviciu (PaaS). Capacitatea furnizată consumatorului cloud este de a implementa aplicaţii folosind limbaje de programare, biblioteci, servicii şi instrumente susţinute de către furnizorul cloud. Un exemplu PaaS este Google App Engine.
- Software ca Serviciu (SaaS). Capacitatea furnizată consumatorului este de a utiliza aplicaţii oferite de către furnizorul de servicii cloud, acestea rulând pe infrastructura cloud a acestuia. Exemple în acest sens sunt Salesforce CRM (Customer Relationship Management) şi Gmail.

Utilizatorii cloud pot alege între diferitele tipuri de servicii cloud, în funcţie de propriile nevoi de portabilitate şi automatizare. De exemplu, în cadrul PaaS se oferă posibilitatea de configurare mai rapidă a aplicaţiilor decât în cadrul IaaS, iar utilizatorii pot exploata oportunităţile de găzduire gratuită oferite de unii furnizori de soluţii PaaS (de exemplu, Google App Engine). Serviciile de tip PaaS şi SaaS oferite de către un furnizor cloud sunt de obicei implementate pe baza unei infrastructuri de tip IaaS, ceea ce face ca interoperabilitatea nivelului IaaS să fie mult mai importantă decât a celorlaltor două niveluri superioare. Interoperabilitatea PaaS depinde în cea mai mare parte de mediile de dezvoltare (de exemplu, Django, Ruby pe Rail, PHP, etc), ce pot fi mai bine susţinute de către comunităţile proprii de dezvoltatori atunci când este necesar. Nivelul SaaS oferă un cadru complet pentru un serviciu utilizator, astfel că principala problemă de interoperabilitate este în mare parte legată de accesul la date.

## Sisteme de management cloud care suportă interoperabilitatea

Implementările reprezentative, la nivel de referinţă pentru sistemele de management cloud de tip IaaS sunt OpenStack, OpenNebula şi Eucalyptus. În baza practicii actuale, aproape toate implementările reprezentative de platforme de management cloud de tip IaaS încearcă să asigure interoperabilitatea cu platforma dominantă, respectiv Amazon EC2.

### OpenStack

OpenStack este un proiect open-source administrat de către fundaţia omonimă. OpenStack este o platformă de management cloud de tip IaaS ce se bazează pe o suită de servicii de bază (Fig. 8):

- Controllerul sistemului de calcul, numit Nova, oferă servere virtuale la cerere. Arhitectura Nova este concepută pentru a putea fi scalată orizontal utilizând echipamente standard, fără cerinţe hardware sau software proprietare, şi cu capacitatea de a se integra cu sistemele existente, precum şi cu alte sisteme de tip IaaS. În ceea ce priveşte hipervizorul acceptat, Nova poate să utilizeze Xen, KVM, LXC şi chiar Microsoft Hyper-V. Nova suportă, de asemenea, diferite arhitecturi (de exemplu, x86, amd64 şi ARM). Acestea fac ca Nova să fie interoperabil cu alte servicii similare din infrastructuri IaaS, cum ar fi Amazon EC2.
- Sistemele de stocare, Swift şi Cinder, asigură stocarea la nivel de obiect şi respectiv dispozitiv de tip bloc. Acestea sunt concepute pentru a fi interoperabile cu sistemele de stocare similare de la Amazon S3 şi EBS, în special Swift, care implementează un subset al API-ului S3 cu ajutorul middleware-ului WSGI.

- Serviciul de stocare imagini, Glance, oferă un catalog de imagini ale discurilor virtuale, şi care include multe caracteristici ale serviciului Amazon AMI (Amazon Machine Imagine). Aceste imagini de disc sunt utilizate în mod obişnuit de către modulul OpenStack Compute. Deşi AMI-urile de la Amazon încă nu pot fi utilizate direct de către serviciul Glance, elementele din API care sunt comune permit unor clienţi terţi de management IaaS, cum ar fi ca exemplu RightScale să automatizeze managementul imaginilor în cadrul OpenStack (folosind propriul format de imagine, compatibil cu AMI).

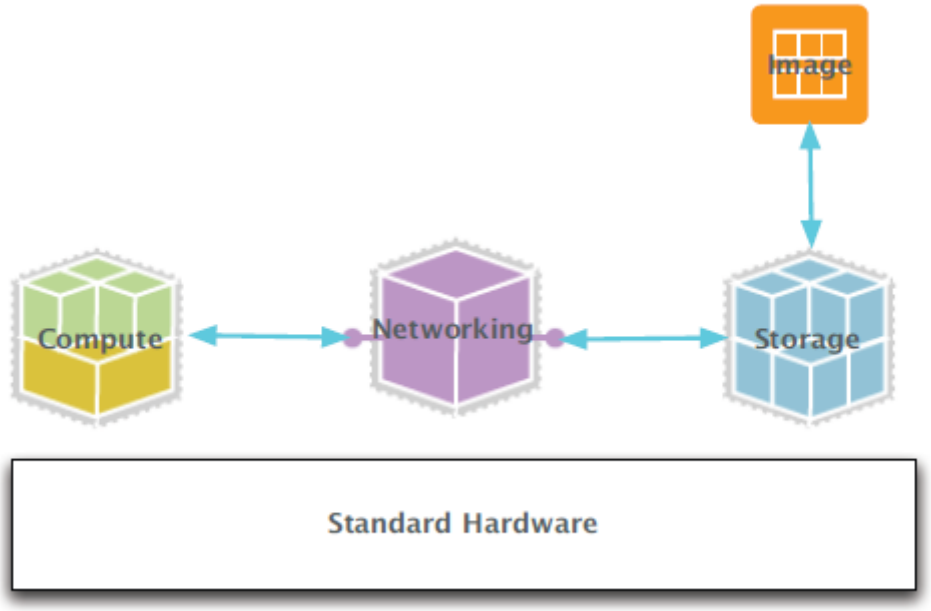



Fig. 8. Arhitectura OpenStack [94]

Sistemul OpenStack pentru reţelistică, Quantum, gestionează configurarea reţelelor şi a adresele IP în cloud. Serviciul Quantum facilitează capacitatea de reţea definită de software în cadrul OpenStack, ce reprezintă o soluţie eficientă pentru implementarea nivelului de reţea la nivel WAN.

## Eucalyptus

Eucalyptus, creat de Eucalyptus Systems, Inc, este o platformă software care creează infrastructuri cloud private şi hibride scalabile în cadrul unei infrastructuri IT existente. Acesta oferă un API care reproduce cu mare fidelitate API-ul Amazon AWS (Amazon Web Services), cu scopul de a interopera cu cloud-ul AWS.

Principalele componente ale platformei Eucalyptus sunt (Fig. 9):

- Cloud Controller, care oferă o funcţionalitate compatibilă cu Amazon EC2;
- Walrus, un sistem de stocare care asigură o funcţionalitate compatibilă Amazon S3;
- Cluster Controller, care gestionează un cluster în cloud;
- Storage Controller, care asigură o funcţionalitate compatibilă Amazon EBS;

- Node Controller, care controlează instanţele maşinilor virtuale.

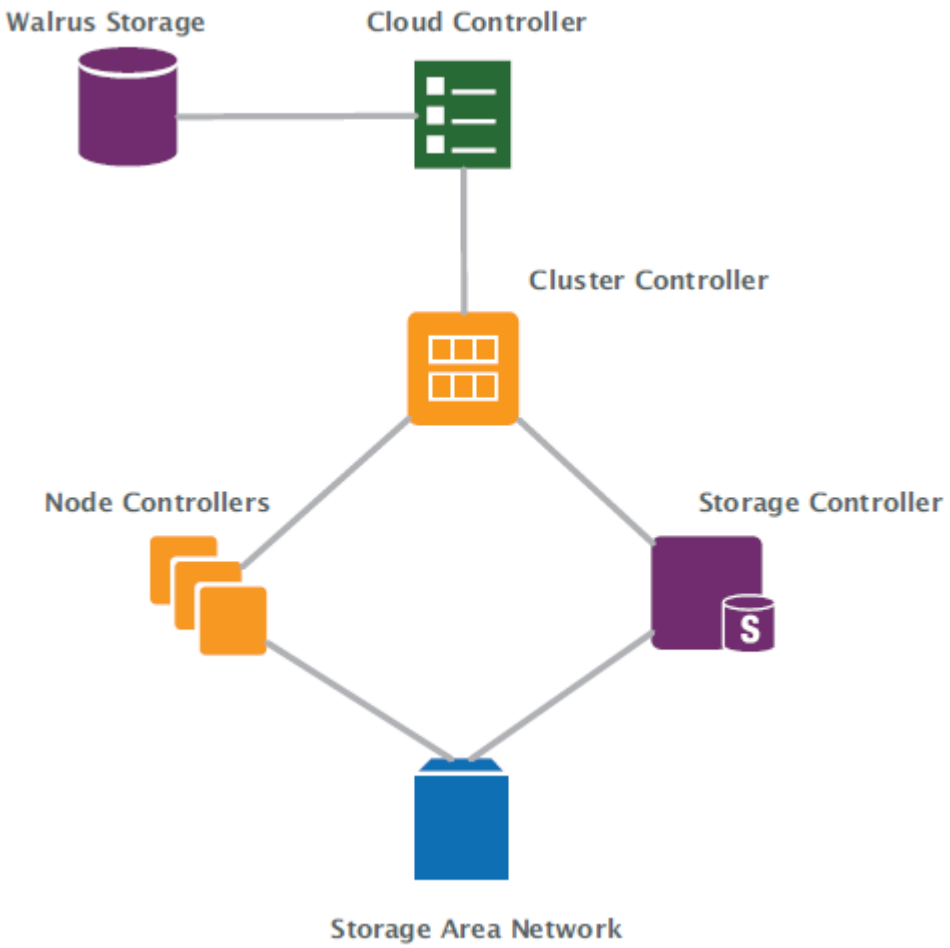



Fig. 9. Arhitectura Eucalyptus [94]

Platforma Eucalyptus oferă flexibilitate în alegerea formatului maşinii virtuale utilizate astfel încât se pot executa atât imagini cu sisteme Windows cât şi Linux, şi se poate construi o bibliotecă de imagini ale maşinilor virtuale care să fie compatibilă cu formatul AMI. De asemenea, imaginile VMware şi vApps pot fi uşor modificate pentru a rula pe Eucalyptus. Utilizatorii cloud au libertatea de a folosi diverse modele de hipervizoare, Eucalyptus fiind compatibil cu vSphere, ESX, KVM şi Xen. Componenta de management a identităţii utilizatorilor poate fi integrată cu sisteme Microsoft Systems Directory sau LDAP. În plus, această componentă este compatibilă cu API-ul AWS Identity and Access Management (IAM), permiţând managementul integrat al infrastructurilor cloud hibride Eucalytus şi AWS.

### OpenNebula

OpenNebula este o platformă open-source ce oferă soluţii flexibile pentru managementul complet al centrelor de date virtualizate, şi permite crearea de infrastructuri cloud private. OpenNebula posedă interfeţe diferite care pot fi folosite pentru a interacţiona cu, şi

gestiona resursele fizice şi virtuale. Principale moduri de de interacţionare cu OpenNebula sunt (Fig. 10):

- Interfeţe cloud pentru consumatorii cloud, ce includ API-uri precum OCCI, EC2 şi interfaţa EBS, şi un portal self-service;
- Interfeţe de administrare pentru administratorii cloud, utilizând linia de comandă şi o interfaţă grafică foarte flexibilă, denumită Sunstone;
- Un API extensibil pentru diferite limbaje de programare precum Ruby, Java;
- Un catalog de aplicaţii care sunt gata de utilizat în cadrul unei instanţe OpenNebula.

În ceea ce priveşte interoperabilitatea cu alte infrastructuri cloud, OpenNebula implementează API-ul EC2 Query care permite integrarea cu infrastructura cloud Amazon EC2, dar şi o interfaţă bazată pe standardul OCCI care este deschis şi public. Prin utilizarea interfeţelor EC2 Query şi OCCI se înlesneşte implementarea de infrastructuri cloud de mari dimensiuni ce sunt deschise către un public larg.

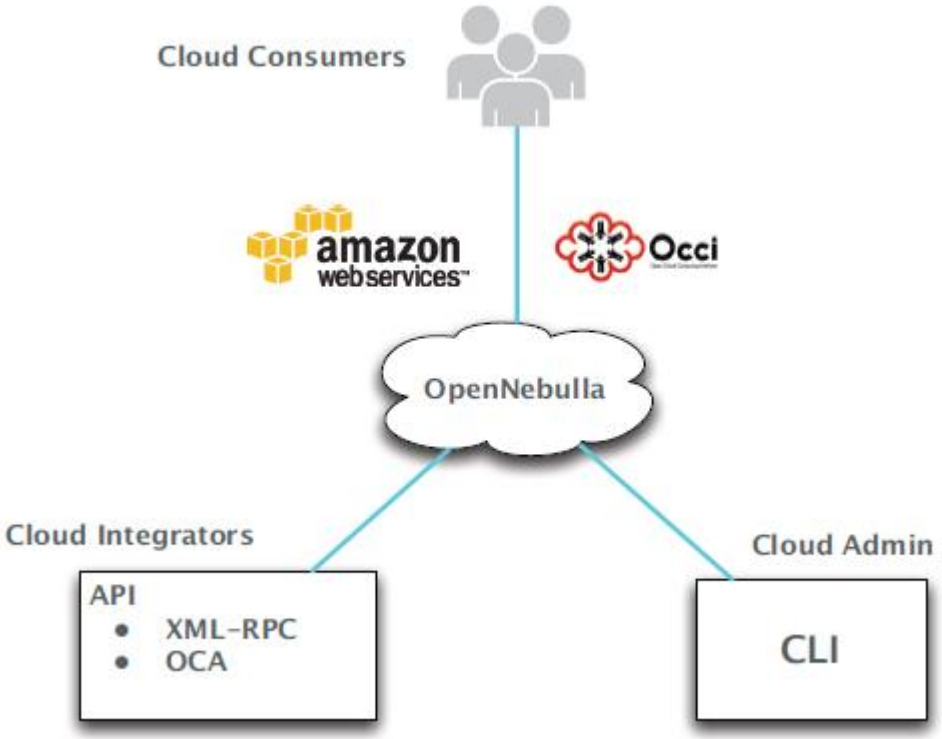



Fig. 10. Arhitectura OpenNebula [94]

Administrarea unei infrastructuri cloud de mari dimensiuni reprezintă o provocare semnificativă din punct de vedere al interoperabilităţii între resurse. Din acest motiv, OpenNebula implementează două tehnologii specifice, respectiv oZones şi VDCs (centre de date virtuale):

- oZones permite managementul centralizat al mai multor instanţe OpenNebula care

sunt numite zone. Aceasta permite administratorilor oZone să monitorizeze fiecare zonă, şi să acorde acces la zone diferite pentru anumiţi utilizatori, utilizând pentru aceasta o interfaţă CLI şi o interfaţă grafică web;

- în interiorul fiecărei zone este posibil să se definească mai multe VDCs care conţin un set de resurse virtuale (imagini, machete de maşini virtuale, reţele virtuale şi maşini virtuale) şi utilizatorii care folosesc şi controlează aceste resurse virtuale.

## 3.4. Virtualizarea sistemului de operare prin intermediul containerelor Docker

Docker este un instrument open-source care automatizează lansarea aplicaţiilor în cadrul containerelor software prin introducerea unui nivel de abstractizare la nivelul virtualizării sistemului de operare. Docker utilizează facilităţi de izolare a resurselor ce sunt prezente în cadrul nucleului (kernel) Linux, precum cgroups şi namespaces, ce permit unor containere independente să fie executate în cadrul unei singure instanţe a unui sistem de operare Linux. Aceasta reduce consumul de resurse şi creşte performanţa în comparaţie cu utilizarea maşinilor virtuale. Componenta namespaces oferă posibilitatea ca fiecare aplicaţie să aibă propria imagine asupra mediului în cadrul sistemului de operare, inclusiv a listei de procese, a id-urilor utilizatorilor, a stivei de protocoale TCP/IP sau a sistemelor de fişiere montate, în timp ce componenta cgroups oferă izolarea resurselor computaţionale precum CPU, memorie, subsistem I/O ce include accesul la disc şi reţea.

Docker include biblioteca libcontainer ca implementare de referinţă pentru utilizarea containerelor. Aceasta este construită pe baza componentelor libvirt, LXC şi systemd-nspawn ce furnizează interfeţe pentru facilităţile oferite de către nucleul Linux mai sus menţionate.

Docker implementează un API de nivel înalt ce oferă suport pentru crearea de containere ce execută procese izolate. Utilizând cgroups şi namespaces, un container Docker, spre deosebire de o maşină virtuală obişnuită, nu necesită sau nu include un sistem de operare propriu. În schimb acesta se bazează pe funcţionalitatea oferită de nucleu, ce este accesat prin intermediul bibliotecii libcontainer, sau a componentelor libvirt, LXC sau systemd-nspawn (Fig. 11).

Prin utilizarea containerelor, resursele sunt izolate, serviciile pot fi restricţionate iar procesele pot fi create cu o vedere proprie asupra sistemului de operare. Acesta include propriul spaţiu de procese şi propria structură de fişiere sau de interfeţe de reţea. Mai multe containere pot să acceseze acelaşi nucleu, dar fiecare container poate fi izolat şi constrâns să utilizeze o cotă de resurse precum CPU, memorie şi I/O. Prin utilizarea Docker pentru crearea şi administrarea containerelor, activitatea de implementare a sistemelor distribuite devine mult mai simplu de realizat. Astfel, este posibil ca mai multe aplicaţii şi procese care au sarcina de a executa anumite activităţi să fie rulate în mod independent în cadrul unei maşini fizice sau a unui grup de maşini virtuale. Aceasta permite lansarea în execuţie

a sistemelor în funcţie de disponibilitatea resurselor sau a necesarului de putere de calcul. Prin acest model de execuţie se pot implementa infrastructuri de calcul cloud de tip PaaS şi se pot scala aplicaţii precum Apache Cassandra, MongoDB sau Riak. De asemenea, se simplifică crearea şi operarea cozilor de execuţie pentru job-uri sau a altor sisteme distribuite.

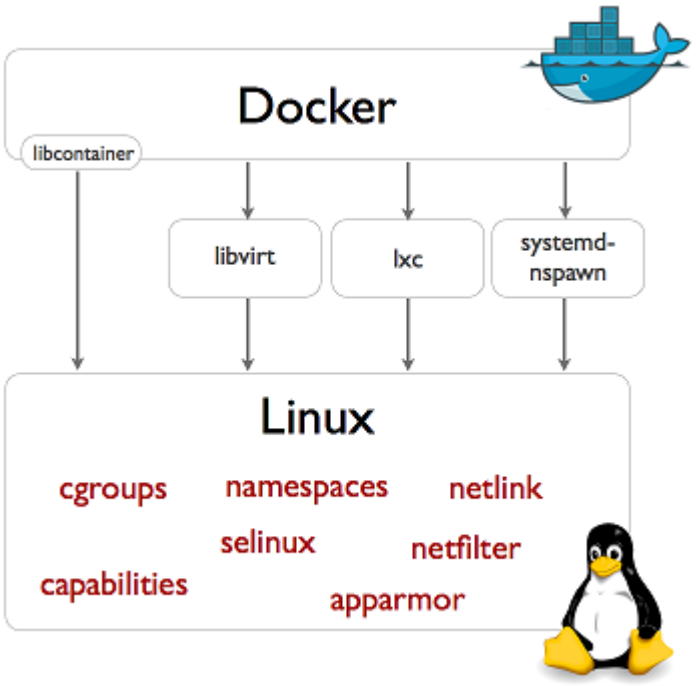



Fig.11. Arhitectura Docker [97]

Docker poate fi integrat în cadrul mai multor instrumente de administrare a infrastructurilor de calcul, inclusiv în cadrul unor platforme cloud publice. Spre exemplu, Docker poate fi utilizat împreună cu Amazon Web Services (AWS), Microsoft Azure, OpenStack Nova, Vagrant, Puppet, Chef, CFEngine, Salt, Ansible, Jenkins, etc. [Wikipedia, Docker].

LXC reprezintă o facilitate de virtualizare a sistemului de operare prin care se pot crea medii de execuţie izolate şi independente, fără a fi necesară utilizarea maşinilor virtuale ce implică un anumit grad de penalizare în ceea ce priveşte performanţa [98]. Docker extinde tehnologia LXC, făcând să devină mult mai accesibilă pentru utilizatori, oferind posibilitatea creării de versiuni, distribuirea şi lansarea în execuţie a sistemelor virtuale. Costul utilizării Docker în termeni de consum al resurselor este mult mai mic decât în cazul utilizării maşinilor virtuale, deoarece acesta nu emulează sistemul de operare complet, ci doar bibliotecile şi fişierele binare ale aplicaţiilor virtualizate (Fig. 12).

Un container Docker se execută în cadrul unui mediu complet virtualizat, astfel încât un algoritm poate fi implementat în orice limbaj care este suportat de sistemul Linux (ex. R, Python, Matlab, C, etc.) şi poate fi compilat utilizând orice bibliotecă compatibilă cu sistemul Linux, fără a cauza conflicte din cauza versiunilor diferite utilizate de către alte aplicaţii. În plus, un container Docker poate fi exportat şi arhivat, şi poate fi executat

indiferent dacă mediul în care va fi rulat este sau nu la fel cu cel iniţial în care a fost implementat.

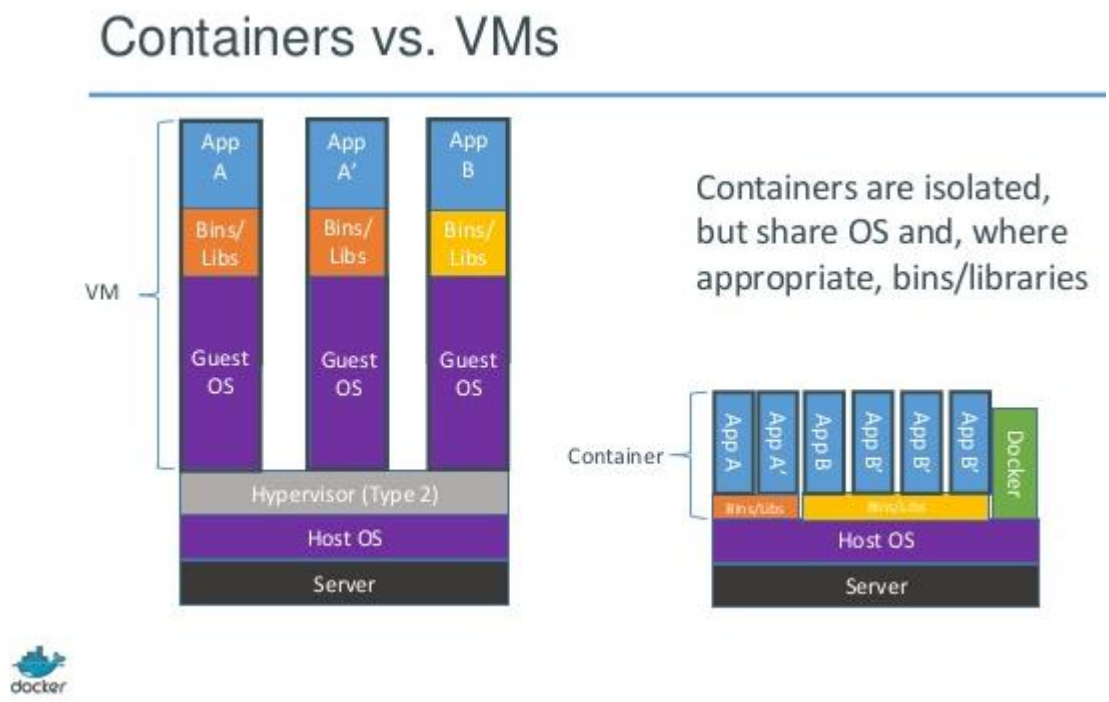



Fig. 12. Comparare între containere şi maşini virtuale [99]

În afară de facilităţile deja menţionate, respectiv furnizarea de medii computaţionale complete şi izolate, Docker mai oferă şi alte funcţionalităţi care îl fac să fie extrem de atractiv pentru implementarea aplicaţiilor ştiinţifice:

- Containerele Docker pot fi implementate cu ajutorul unui script obişnuit, astfel încât acestea pot avea mai multe versiuni, pot fi partajate, şi pot fi recreate cu un simplu fişier text (aşa numitul fişier Dockerfile).
- Containerul Docker poate avea mai multe versiuni şi poate fi modificat în paralel în mai multe ramuri, similar cu un repository în care se păstrează versiunile codului sursă al unui program, spre exemplu un repository Git. Aceste versiuni pot fi partajate în mod direct utilizând repository-ul administrat chiar de către Docker – Docker Hub.
- Spre deosebire de o maşină virtuală, un container Docker îşi poate salva şi restabili conţinutul original de fiecare dată când este lansat în execuţie, asigurând consistenţa mediului computaţional.
- În cadrul unui container pot fi încapsulate şi alte fişiere precum date sau documentaţie, astfel încât poate fi utilizată o singură arhivă pentru partajarea întregului mediu de execuţie a aplicaţiei.
- Există o comunitate foarte mare de utilizatori, în special grupurile dedicate dezvoltării de instrumente operaţionale (DevOps) şi cel al dezvoltării de aplicaţii web, care pot oferi informaţii specifice pentru un anumit caz de utilizare sau suport

pentru rezolvarea problemelor.

- Containerele Docker gestionează doar resursele pe care le utilează curent, astfel încât penalizarea performanţei este minimă în comparaţie cu o aplicaţie nativă, spre deosebire de o maşină virtuală ce trebuie să aibă pre-alocate anumite resurse, precum CPU şi memorie, pentru întreaga perioadă de execuţie.
- Directoarele din cadrul sistemului gazdă pot fi montate cu uşurinţă în cadrul unui sistem de fişiere Docker, astfel încât codul sursă şi datele unei aplicaţii pot fi partajate comod între cele două sisteme.

În continuare se vor exemplifica câteva din beneficiile tehnologiei Docker [100].

**Utilizarea imaginilor Docker rezolvă problema dependenţelor între diverse versiuni de aplicaţii şi biblioteci.**

Abordarea în cazul Docker este similară cu cea în cazul utilizării unei maşini virtuale, în sensul că se rezolvă problema dependenţelor între diverse versiuni de aplicaţii şi biblioteci prin utilizarea unei singure imagini binare în care toate componentele software necesare sunt pre-instalate, configurate şi testate. O diferenţă esenţială între imaginile Docker şi imaginile maşinilor virtuale constă în faptul că imaginile Docker folosesc acelaşi nucleu Linux cu sistemul gazdă. Din punctul de vedere al utilizatorului aceasta înseamnă că o imagine Docker trebuie să fie bazată pe sistemul de operare Linux şi să folosească aplicaţii compatibile Linux. Partajarea nucleului Linux cu sistemul gazdă face ca Docker să fie mai eficient, din punct de vedere al consumului de resurse, ca o maşină virtuală. Spre exemplu, un sistem gazdă de tip desktop poate să execute câteva maşini virtuale simultan, însă nu are probleme în a executa sute de containere Docker – un container reprezintă instanţa în execuţie a unei imagini. Din acest motiv, Docker este extrem de popular şi atractiv pentru utilizatorii şi furnizorii de servicii cloud.

**Fişierele Dockerfile pot suplini lipsa unei documentaţii clare în ceea ce priveşte dependenţele unei aplicaţii.**

Cu toate că imaginile Docker pot fi create în mod interactiv, această modalitate de lucru îngreunează înregistrarea transparentă a acţiunilor întreprinse, astfel încât acestea să poată fi repetate când este necesar. Un fişier Dockerfile conţine un script similar cu un fişier de tip Makefile în care este definit exact modul în care imaginea a fost construită. Sintaxa fişierelor Dockerfile este mai simplă decât a altor instrumente utilizate pentru managementul configuraţiei, precum Chef, Puppet sau Ansible, sau a unor instrumente utilizate pentru integrarea componentelor software ale unui proiect, precum Travis CI. Utilizatorii nu trebuie decât să fie familiarizaţi cu programarea script-urilor shell şi cu mediul de lucru al distribuţiilor Linux (de pildă, utilizarea instrumentului apt-get în cadrul distribuţiilor bazate pe Debian). Acest mod de lucru are numeroase avantaje, astfel:

- Chiar dacă imaginea unui sistem devine foarte mare (de ordinul gigabytes), un fişier Dockerfile este doar un simplu script care poate să fie stocat şi partajat cu

uşurinţă.

- Fişierele de tip text sunt ideale pentru a fi utilizate în cadrul unui sistem de management al versiunilor precum Subversion sau Git, acestea fiind capabile să urmărească toate modificările efectuate în cadrul fişierului Dockerfile.
- Fişierul Dockerfile oferă o imagine de ansamblu extrem de eficientă care rezumă toate dependenţele software, variabilele de mediu utilizate, şi alte elemente necesare pentru execuţia aplicaţiei. Dependenţele unei aplicaţii sunt mult mai uşor de evidenţiat în acest fel, suplinind efortul de realizare a unei documentaţii finale în care să fie menţionate toate dependenţele aplicaţiei. Aceste sunt efectiv menţionate în momentul realizării fişierului Dockerfile.
- Spre deosebire de un fişier de tip Makefile, un fişier Dockerfile include toate dependenţele software, inclusiv la nivel de sistem de operare. Acesta este compilat cu ajutorului unui instrument specific, astfel încât posibilitatea ca versiunea executabilă să difere în funcţie de sistemul gazdă pe care trebuie să ruleze este puţin probabilă.
- Este posibil să fie adăugate teste şi să se facă verificări ale comenzilor utilizate pentru instalarea aplicaţiilor, astfel încât să se asigure faptul că procesul de instalare şi configurare al componentelor software dintr-o imagine s-a terminat cu succes.
- Utilizatorii pot extinde şi personaliza cu uşurinţă imaginile create de alţi utilizatori, prin simpla editare a fişierului Dockerfile.

**Utilizarea Docker ca mediu local de dezvoltare**

O cerinţă foarte importantă în cadrul activităţii de cercetare constă în posibilitatea de reproducere a rezultatelor unui experiment. De asemenea, este foarte important ca un instrument software să poată fi utilizat cu uşurinţă şi să poată fi integrat în cadrul unui flux de activităţi de cercetare. Din acest motiv, se poate explica interesul scăzut acordat unor tehnologii similare care au fost propuse în trecut. Orice instrument nou care nu este foarte familiar utilizatorilor poate întâmpina o rezistenţă în adoptarea ca soluţie standard. Totuşi, Docker propune câteva elemente inovative, care sunt atât utile, cât şi uşor de aplicat în cadrul activităţilor curente, astfel încât efortul de migrare sau de portare a unei aplicaţii native în cadrul containerelor Docker este minim.

În timp ce susţinătorii utilizării maşinilor virtuale pentru aplicaţiile native menţionează faptul că acestea sunt disponibile exclusiv în cadrul infrastructurilor cloud, foarte mulţi cercetători lucrează pe sisteme locale, respectiv utilizează calculatoarele portabile sau desktop-urile în care au instalate toate aplicaţiile necesare. Nevoia de utilizare a serviciilor cloud poate să apară doar pentru anumite activităţi cu caracter colaborativ sau atunci când rezultatele obţinute sunt suficient de satisfăcătoare astfel încât să fie pusă problema unei necesar suplimentar de putere calcul. În cadrul procesului de dezvoltare local, un cercetător poate să utilizeze diverse aplicaţii pentru editarea fişierelor, compilarea şi depanarea codului sursă a unei aplicaţii, medii de dezvoltare integrate sau sisteme de

control al versiunilor unui program. Calculul ştiinţific ce utilizează sisteme la distanţă, prin contrast, se bazează pe interacţiunea cu un mediu mai puţin familiar, în marea majoritate a cazurilor în mod text şi în linie de comandă.

Docker poate fi instalat cu uşurinţă pe majoritatea platformelor de calcul. În cadrul sistemelor de operare care nu se bazează pe nucleul Linux, precum Mac şi Windows, Docker poate fi instalat cu ajutorul unei maşini virtuale de tip VirtualBox, şi apoi poate fi executat local. Docker permite utilizatorului să asocieze orice director, precum directorul curent pentru un anumit program sau proiect, cu un container Docker aflat în execuţie. Aceasta permite unui utilizator să beneficieze în continuare de familiaritatea oferită de sistemul de operare local, în cazul efectuării operaţiunilor uzuale, cum ar fi: editarea de fişiere, navigare pe Internet, control al versiunilor, iar execuţia codului să se petreacă în cadrul mediului creat în mod controlat de către Docker, prin instanţierea unei imagini şi rularea aplicaţiei în cadrul unui container.

**Portabilitatea şi partajarea aplicaţiilor**

Un avantaj imediat al utilizării Docker constă în faptul că mediul de execuţie al unei aplicaţii este portabil. Containerele LXC nu oferă această garanţie a păstrării mediului în care se execută aplicaţiile pe care le conţin din diverse motive: configuraţii diferite ale sistemului de stocare, ale stivei de protocoale TCP/IP, politici diferite de audit, etc. Pe de altă parte, Docker gestionează distribuirea şi execuţia unui container astfel încât acesta rulează în acelaşi mediu indiferent de sistemul gazdă şi expune acelaşi set de interfeţe de reţea sau de stocare. Această facilitate este extrem de utilă pentru capacitatea de reproducere a execuţiei unei aplicaţii sau a unui experiment, în situaţia în care un utilizator doreşte să recreeze mediul de execuţie al unei aplicaţii care a fost dezvoltată de către o terţă persoană. Spre exemplu, un cercetător ar dori să execute un algoritm în cadrul unui server cloud, care are mai multă memorie şi mai multă putere de calcul decât sistemul local, sau ar dori să depaneze o anumită problemă. În ambele situaţii, utilizatorul poate să exporte o imagine a containerului aflat în execuţie:

*docker export container-name > container.tar*

şi apoi poate să ruleze un container similar în cadrul serverului cloud.

Partajarea imaginilor este facilitată de către infrastructura Docker Hub. Imaginile Docker tind să aibă o dimensiune mai mică decât a maşinilor virtuale echivalente. Copierea unei imagini mari poate să dureze foarte mult şi să devină astfel o problemă. Serviciul Docker Hub este disponibil în mod public tuturor utilizatorilor înregistraţi, şi, de asemenea, este disponibil ca aplicaţie open-source care poate fi implementată local în cadrul propriei infrastructuri. Spre exemplu, partajarea unei imagini publice se poate face utilizând comanda Docker *push,* urmată de numele imaginii respective:

*docker push username/r-recommended*

Dacă un fişier Dockerfile este disponibil în cadrul unui repository public precum Github sau Bitbucket, Docker Hub poate să utilizeze acel fişier pentru construirea automată a imaginii. Acest lucru se poate realiza de fiecare dată când se modifică fişierul Dockerfile, astfel încât comanda *push* poate să nu mai fie necesară. Un utilizator poate să actualizeze varianta locală a imaginii utilizând comanda *pull.* Aceasta descarcă toate modificările care au fost făcute în cadrul imaginii publicate în Docker Hub.

**Reutilizarea modulelor**

Modul de abordare al Docker oferă o soluţie tehnică pentru o problemă comună în cadrul modelului bazat pe utilizarea maşinilor virtuale: reutilizarea şi combinarea elementelor componente. Acest lucru se realizează prin utilizarea fişierelor Dockerfile, prin care se creează o descriere programatică a mediului de execuţie al aplicaţiei, ce poate fi configurat în funcţie de necesităţi. În plus, Docker are şi o caracteristică proprie: facilitează reutilizarea componentelor modulare prin construirea unui container pe suportul unui alt container. În loc ca fişierul Dockerfile să fie copiat şi apoi să fie adăugate mai multe componente la final, se poate declara că un nou fişier Dockerfile este construit pe baza unui fişier Dockerfile anterior, utilizând directiva FROM. Spre exemplu, într-un fişier Dockerfile se adaugă mediul de execuţie pentru calcule statistice R în cadrul unei distribuţii Ubuntu Linux:

*FROM ubuntu:latest*
*RUN apt-get update*
*RUN apt-get -y install r-recommended*

Aceste comenzi sunt aproximativ similare cu declararea dependenţelor pentru o aplicaţie software. Spre deosebire însă de modul în care se declară dependenţele pentru aplicaţiile native, un fişier Dockerfile poate avea doar o singură dependenţă (o singură linie *FROM*). Acest fişier Dockerfile poate fi apoi compilat prin executarea în directorul curent a următoarei comenzi:

*docker build -t username/r-recommended*

Dacă această imagine este partajată (fie direct, fie prin intermediul infrastructurii Docker Hub), atunci un alt utilizator va putea construi un alt container pe baza acestei imagini, în loc să utilizeze imaginea de bază *ubuntu:latest* ce a fost declarată în cadrul primului fişier Dockerfile. Aceasta se face prin utilizarea directivei *FROM* în cadrul fişierului Dockerfile. Spre exemplu:

*FROM username/r-recommended*
*RUN apt-get update*
*RUN apt-get -y install r-cran-matrix*

Deşi acest exemplu prezintă doar un simplu caz în care există o singură dependenţă în fiecare nivel, prin acest procedeu extrem de flexibil se pot crea medii de execuţie foarte complexe.

Fiecare nivel reprezintă o parte componentă ce oferă doar ceea ce este necesar pentru a executa un anumit serviciu, iar interfeţele sunt strict dedicate conectării mai multor niveluri. Spre exemplu, un container poate rula serviciul PostgreSQL, ce oferă acces la date pentru un alt container, în care rulează un algoritm care analizează datele din baza de date PostgreSQL:

*docker run -d --name db training/postgres*
*docker run -d -P --link db:db training/webapp python app.py*

Faptul că un singur sistem desktop obişnuit poate să execute cu uşurinţă un număr de containere de ordinul sutelor permite descompunerea arhitecturilor complexe în elemente logice compacte ce pot fi reutilizate şi recreate identic într-un alt context. Spre exemplu, un utilizator poate să creeze un container ce oferă mediul computaţional necesar pentru implementarea algoritmului de analiză a datelor într-un alt limbaj de programare, şi apoi să îl conecteze cu acelaşi container iniţial ce oferă baza de date PostgreSQL.

**Utilizarea versiunilor pentru containere**

În plus faţă de posibilitatea creării de versiuni pentru fişierul Dockerfile, este posibilă crearea de versiuni şi pentru imagini. Imaginile Docker, la fel ca şi containerele, conţin elemente de tip metadata ce specifică data, autorul, imaginea părinte şi alte detalii importante pentru identificare. O imagine poate fi restaurată la o versiune anterioară, iar modificările realizate în cadrul unui container pot fi corectate prin revenirea la o stare iniţială sau o versiune anume. Modificările făcute în cadrul imaginii *ubuntu:latest* pot fi inspectate cu ajutorul următoarei comenzi:

*docker history ubuntu:latest*

Se poate identifica astfel o versiune anterioară, la care se doreşte să se revină, prin ajustarea elementului *tag*, corespunzător cu valoarea hash-ului asociat imaginii respective. Spre exemplu:

*docker tag 25f ubuntu:latest*

Pe lângă această facilitate, Docker oferă şi posibilitatea de a încărcă şi descărca incremental imaginile prin transferarea numai a diferenţelor între două imagini, în loc ca acestea să fie transferate complet.

**Cele mai bune practici pentru utilizarea containerelor Docker**

- Este important să se utilizeze containerele Docker în etapa de dezvoltare a unui proiect software. Un avantaj al Docker constă în faptul că acesta reproduce cât mai fidel modul în care un utilizator dezvoltă o aplicaţie. Codul care se execută în cadrul unui container pe un sistem local este identic cu cel care se execută nativ, dar are avantajul că poate fi transferat cu uşurinţă într-un mediu de execuţie similar, dar care beneficiază de resurse computaţionale mai puternice.

- Este de preferat să se utilizeze fişierele Dockerfile în locul sesiunilor interactive. Deşi Docker poate fi utilizat într-o manieră interactivă, acest procedeu nu este foarte eficient în situaţia în care se doreşte refacerea unui experiment.
- Este foarte util să se adauge teste şi să se facă verificări ale fişierului Dockerfile. Comenzile prezente în cadrul unui fişier Dockerfile nu se limitează doar la instalarea de aplicaţii software, ci pot include şi execuţia unor alte aplicaţii. Astfel se poate verifica dacă o imagine a fost construită corect şi dacă ea conţine toate componenentele software necesare pentru aplicaţia de interes.
- Este bine să se utilizeze şi să se furnizeze imaginile de bază relevante. Cu toate că Docker permite o arhitectură modulară, este în sarcina utilizatorului să profite de această facilitate. Un flux de lucru normal se bazează pe un fişier Dockerfile ce include toate dependenţele software necesare în procesul de dezvoltare şi care pot fi extinse prin utilizarea unei imagini Docker dedicate unui anumit proiect sau aplicaţii. Prin reutilizarea imaginilor existente se reduce efortul necesar pentru implementarea şi configurarea unui mediu de dezvoltare. De asemenea, se promovează efortul de standardizare într-un anumit domeniu şi se beneficiază de facilitatea Docker de a transfera doar diferenţele dintre imagini.
- Este recomandat să se partajeze imaginile Docker şi fişierele Dockerfile. Infrastructura Docker Hub reduce în mod semnificativ efortul necesar pentru distribuirea de imagini ce au o dimensiune foarte mare. Prin publicarea imaginilor, acestea pot să fie accesate apoi şi de către alţi utilizatori.
- Este bine să se folosească arhive şi să se utilizeze replici ale stării curente. Docker oferă facilitatea de creare de versiuni, atât pentru imagini, cât şi pentru fişierele Dockerfile.

**Limitări ale Docker**

Docker are potenţialul necesar rezolvării unor limitări existente în anumite situaţii în care trebuie recreate medii computaţionale complexe. Docker prezintă, de asemenea, mai multe avantaje faţă de alte tehnologii similare. Spre exemplu, posibilitatea de a utiliza versiuni atât pentru imagini, cât şi pentru fişierele Dockerfile, arhitectura modulară, portabilitatea containerelor şi interfeţele simple, au reprezentat factori de succes în adoptarea acestei tehnologii în industrie. Similar, Docker are un potenţial important să devină o tehnologie standard şi în mediul ştiinţific şi de cercetare. Cu toate acestea, există mai multe limitări şi este necesar să se evalueze impactul acestora asupra activităţii în care se doreşte utilizarea Docker:

- Docker nu oferă o soluţie completă de virtualizare, ci se bazează pe nucleul Linux ce este oferit de către sistemul gazdă.
- Docker este limitat la sisteme gazdă cu arhitectură de tip 64 bit, deci nu este disponibil pe sisteme de calcul mai vechi.
- Pe sistemele Mac şi Windows, Docker trebuie să fie executat în cadrul unei maşini

virtuale Linux. Deşi instrumentul boot2docker facilitează acest proces, totuşi, această situaţie poate reprezenta un incovenient.

- Problemele potenţiale de securitate trebuie încă să fie evaluate. Suportul pentru semnarea imaginilor Docker utilizând certificate digitale ar putea simplifica construirea de containere sigure.
- Deocamdată nu există date certe cu privire la gradul de adopţie al Docker în cadrul comunităţii ştiinţifice, şi rămâne de văzut cât de popular va fi acesta în viitor.

# 4. Elaborarea unui cadru arhitectural standardizat comun

## 4.1. Arhitectura sistemului

Progresul eficient şi sustenabil, atât în domeniul controlului proceselor, cât şi al activităţii de cercetare-dezvoltare, poate fi asigurat prin utilizarea unor mijloace adecvate în dezvoltarea de algoritmi integrând reacţiile de la implementările în industrie, formulând problemele legate direct de cazuri reale şi găsind rezolvări la aceste probleme. Cele mai eficiente sisteme automate dedicate pentru controlul proceselor industriale sunt în prezent caracterizate printr-o structură ierarhică presupunând existenţa a cel puţin două niveluri de automatizare: un nivel executiv responsabil de controlul clasic al parametrilor principali de proces şi un nivel de supraveghere, responsabil de monitorizarea instalaţiilor şi de luare a deciziilor. Cele mai multe dintre mediile industriale necesită un sistem cu un nivel de control superior, capabil să îmbunătăţească şi să optimizeze funcţionarea instalaţiei. Paradigma RH Control [101,102] (Fig. 13) impune implementarea unui astfel de nivel, dar realizarea acestuia este dependentă de existenţa unei biblioteci de algoritmi bine documentată şi a unei metodologii riguroase de reprezentare care să faciliteze comunicarea şi schimbul de informaţii. Incorporarea ultimelor cercetări în sfera algoritmilor întăreşte relaţiile între mediul academic şi cercetarea industrială, permiţând transferul de cunoştinte către aplicaţiile industriale şi acoperind golul dintre dezvoltarea teoretică şi implementarea în timp real.

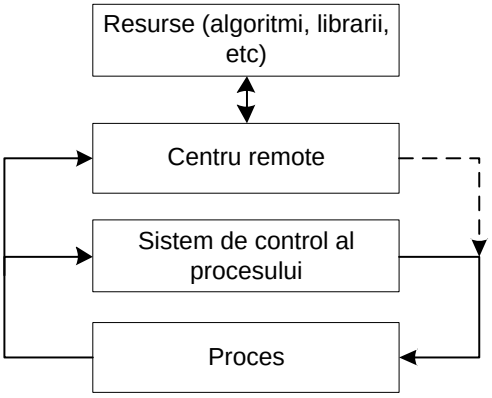



Fig. 13. Structura funcţională

Arhitectura sistemului este prezentată în Fig. 14, în care sunt evidenţiate cele trei componente principale ale acestuia: modulul de dezvoltare şi testare, biblioteca online şi subsistemul de execuţie online. Din punct de vedere structural, sistemul va funcţiona ca o aplicaţie web standard având o bază de date relaţională, care va permite accesul la fişierele cu diverşi algoritmi prin organizarea acestora în funcţie de tipul algoritmilor şi de procesele cărora li se pretează. La această aplicaţie web se adaugă un modul responsabil cu implementarea şi testarea algoritmilor. Acest modul poate folosi FBDK pentru elaborarea algoritmilor, pornind de la un fişier descriptor ce conţine diagrame de execuţie, formule, diagrame de stare etc. Pentru fiecare algoritm se va adăuga o descriere care să definească concret domeniul de utilizare, să detalieze performanţele şi limitările. Dacă este cazul, se va specifica dacă un anumit algoritm foloseşte alte funcţii din bibliotecă, dacă au existat implementări ale acestuia în mediul industrial, caracteristicile procesului şi rezultatele obţinute.

Biblioteca permite două moduri de utilizare a componentelor sale: descărcare pentru a putea fi integrate într-un controller de proces (implementare offline) sau execuţie directă în bibliotecă (execuţie online). În cazul utilizării offline, utilizatorului i se pune la dispoziţie un fişier ce conţine o funcţie bloc simplă sau compusă ce implementează funcţionalitatea dorită. În cazul utilizării online, reprezentarea va fi sub forma unei configuraţii de sistem care să permită rularea şi conectarea la o aplicaţie existentă. Pentru aceasta vor fi dezvoltate funcţii bloc speciale necesare pentru interfaţarea cu procesul. Biblioteca online va fi alcătuită dintr-un sistem de baze de date care va asigura stocarea coerentă a algoritmilor și accesarea lor într-un mod optim, un sistem de asamblare și prezentare a informațiilor, folosit pentru interacțiunea cu exteriorul și validarea introducerii algoritmilor/blocurilor funcționale și un modul de comunicație care va prezenta algoritmii stocați folosind standardul ales (fie el IEC 61499 sau orice alt standard ulterior preferat) astfel încât algoritmii să poată fi implementați în sistemul de control al procesului pentru execuția acestora în mod optimizat. În plus, este necesară existența unui modul de simulare/testare, fie înglobat în bibliotecă, fie extern acesteia, care să se ocupe cu validarea și simularea rulării acestor date, pentru a se putea identifica oportunitatea folosirii unui anume algoritm în situații concrete. Reala valoare a acestui modul constă în culegerea datelor din timpul rulării și integrarea acestora în cadrul informațiilor despre algoritmi, conținute în biblioteca de algoritmi, oferind un istoric util în clasificarea eficienței, împreună cu rata de success/eșec a fiecărui algoritm în parte.

În ceea ce privește implementarea efectivă a bibliotecii, este preferată abordarea modulară, care să permită un nivel crescut de portabilitate și o abstractizare a cadrului în care rulează. Astfel se poate folosi o implementare bazată pe mașini virtuale (VM) integrate în mediul cloud, conținând fiecare toate resursele necesare rulării modulului/ modulelor incluse în acel VM, lucru ce permite portarea bibliotecii între diferiții ofertanți de soluții cloud, fie ele soluții publice, cât și private sau hibride, conform paradigmei PaaS. Alternativa este folosirea containerelor Linux (LXC sau Docker), care permite izolarea resurselor, fie ele CPU, memorie, I/O, rețea, sau chiar a proceselor care rulează pe acea mașină, făcându-se astfel posibilă separarea și portarea fără a mai fi nevoie de folosirea unei mașini virtuale pentru fiecare componentă/proces din bibliotecă. Avantajul acestei soluții este înaltul grad de integrare cu soluțiile cloud deja existente pe piață și flexibilitatea portabilității, inclusiv în timpul funcționării, fără a afecta disponibilitatea serviciului conform paradigmei SaaS.

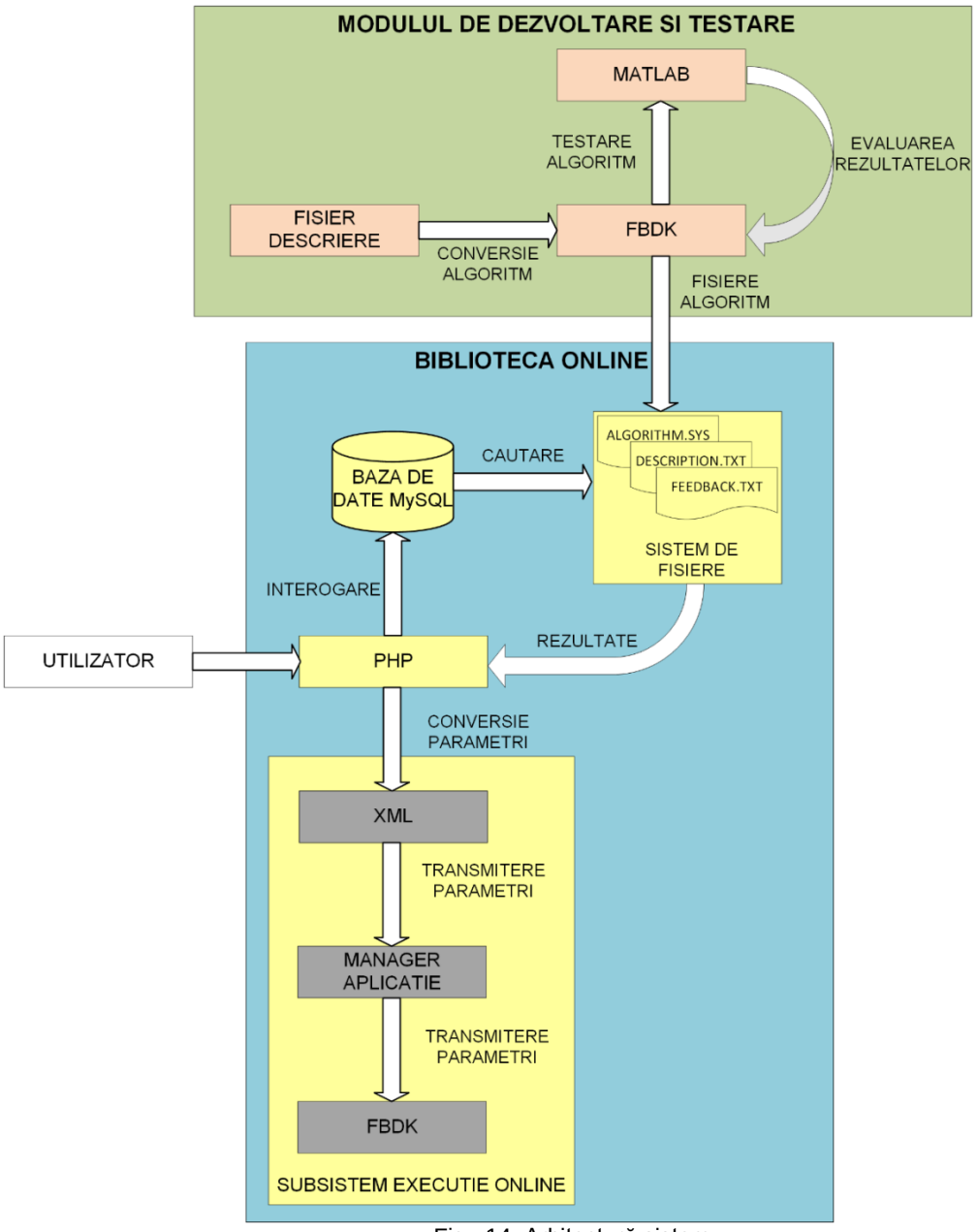



Fig. 14. Arhitectură sistem

Sistemul online trebuie să îndeplinească anumite funcţii:

- Să permită stocarea algoritmilor reutilizabili de control.
- Să permită accesul online la algoritmii disponibili în bibliotecă.
- Să ofere posibilitatea de executare online a unor algoritmi din bliblioteca şi furnizarea rezultatului execuţiei acestora către un controller aflat la distanţă;
- Să pună la dispoziţia administratorilor de sistem mecanismele necesare gestionării înregistrărilor (algoritmi şi utilizatori), astfel încât să nu existe duplicate;
- Să asigurare corectitudinea şi îndeplinirea criteriilor de performanţă ale algoritmilor, prin verificarea şi validarea acestora înainte de a fi disponibili utilizatorilor;
- Să definească limitele de utilizare ale algoritmilor disponibili (tipul de aplicaţie, numărul de variabile de intrare/ieşire etc.).

Biblioteca online va avea patru tipuri de utilizatori: utilizatori neînregistraţi, utilizatori privilegiaţi, utilizatori avansaţi şi administratori de sistem. Cele patru tipuri de utilizatori au un nivel de privilegii diferit, după cum a fost ilustrat în Fig. 15.

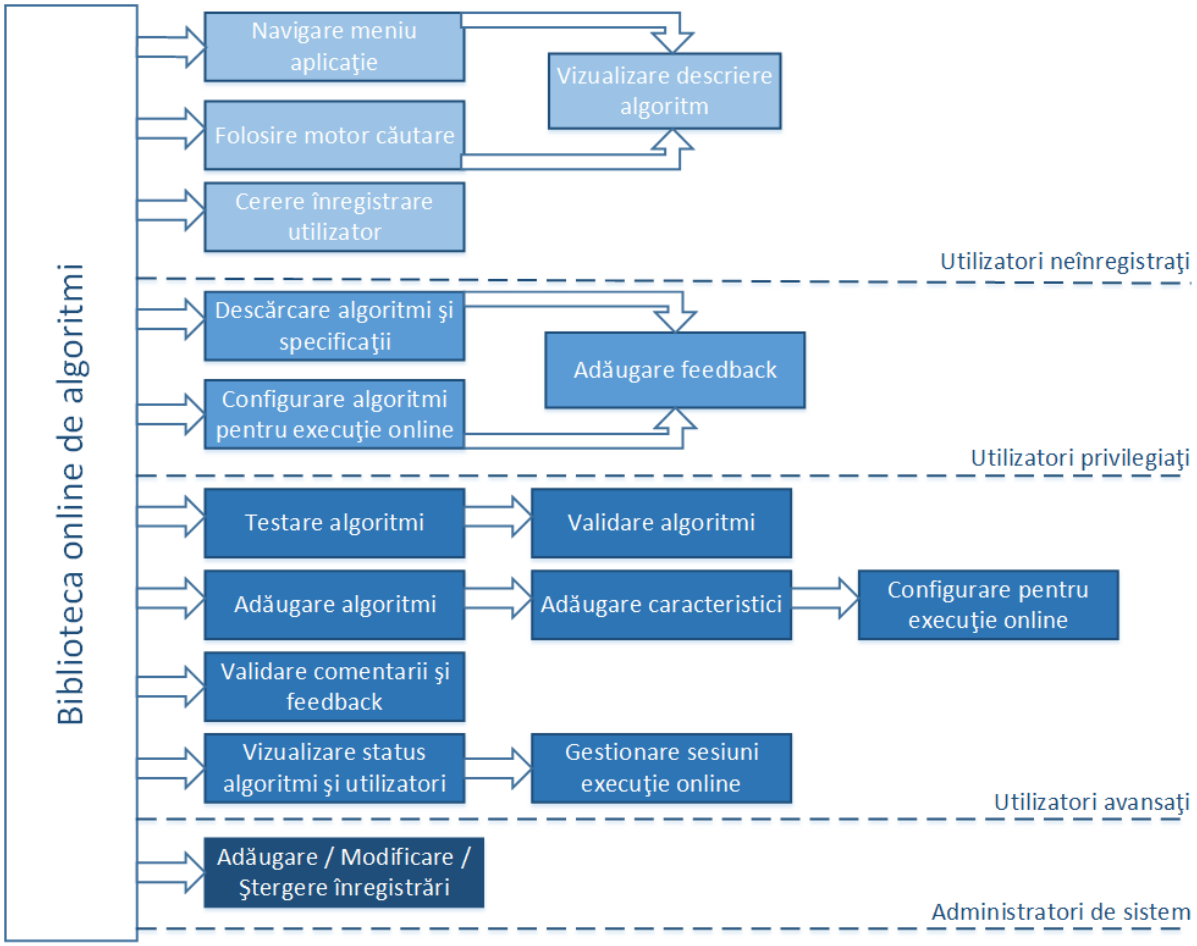



Fig. 15. Funcţiile sistemului

Utilizatorii neînregistraţi au acces la interfaţa de bază a aplicaţiei. Ei pot căuta algoritmi, vizualiza caracteristicile acestora şi eventualele rezultate obţinute în urma implementării lor într-o aplicaţie. Ei se pot înregistra, prin completarea unui formular, pentru a putea avea acces la funcţiile specifice utilizatorilor privilegiaţi. Utilizatorii privilegiaţi pot în plus să descarce algoritmii pentru a-i putea folosi în aplicaţii de control al proceselor, sau pot configura execuţia lor online astfel încât rezultatul să fie trimis la un echipament aflat la distanţă. Ei îşi pot accesa istoricul acţiunilor (ce algoritmi au descărcat, ce feedback sau comentarii au adăugat şi ce algoritmi au avut sau au în execuţie online, precum şi caracteristicile acestora). Aceşti utilizatori pot, de asemenea, introduce comentarii sau fişiere înregistrând răspunsul procesului în care au fost implementaţi. Utilizatorii avansaţi au la dispoziţie aceleaşi funcţii ca şi utilizatorii privilegiaţi şi pot, în plus, să vadă care sunt algoritmii aflaţi în execuţie la acel moment, să vadă istoricul activităţii utilizatorilor privilegiaţi, să adauge algoritmi noi şi să vizualizeze algoritmii adăugaţi care nu au fost încă validaţi. De asemenea, ei pot gestiona algoritmii online, astfel că pot opri execuţia fără a fi necesar acordul utilizatorului privilegiat care o lansase, dar înştiinţându-l de acest lucru. Aceşti utilizatori pot aplica metode de verificare, validare sau optimizare a algoritmilor prin utilizarea unor platforme de simulare. Administratorii bibliotecii au drepturi depline asupra înregistrărilor din baza de date. Ei pot adăuga sau şterge noi

categorii de procese sau algoritmi, pot adăuga sau şterge utilizatori şi sunt responsabili de asigurarea integrităţii bibliotecii şi a componentelor acesteia.

## 4.2. Platforma colaborativă

Aplicaţiile informatice sprijină inginerii şi cercetătorii în îndeplinirea sarcinilor şi obiectivelor, prin:
- obţinerea mai rapidă de informaţii (din surse deschise sau multisursă) prin intermediul unor instrumente automate;
- diseminarea documentaţiei, rapoartelor, analizelor, prin intermediul unor canale de comunicare multiple;
- coordonarea echipelor de cercetare, aproape în timp real, cu ajutorul instrumentelor de colaborare şi fluxurilor informaţionale.

Platformele colaborative oferă cercetătorilor capacitatea de a lucra împreună, de a produce sinergie într-un mediu şi la nivelul relaţiilor ce acţionează în acel mediu. Această tehnologie informaţională oferă un sprijin instrumental analitic menit să permită integrarea informaţiilor din mai multe surse, prin reproiectarea întregului ciclu de viaţă al unui sistem informatic, utilizând tehnologiile web. Construirea unei platforme colaborative ajută realizarea unui management eficient al cunoştinţelor, soluţiile avute în vedere conţinând un grad ridicat de noutate şi flexibilitate, în următorul context: partajarea informaţiilor utile, generarea de idei inovative, promovarea lucrului în echipă, delocalizarea şi descentralizarea activităţilor, valorificarea rezultatelor activităţilor ştiinţifice desfăşurate.

Dezvoltarea societăţii informaţionale ca societate a cunoaşterii este condiţionată de prezenţa unor organizaţii cu capacităţi avansate de gestionare a competenţelor lor colective ca surse de performanţă. Asigurarea fezabilităţii proiectelor de dezvoltare, de către organizaţii bazate pe cunoaştere, presupune conjugarea efortului strategic de informatizare cu o susţinere educaţională şi managerială adecvată. Procesul de cercetare poate fi asimilat teoretic unui proces de producţie, în care uneltele şi mijloacele de fabricaţie sunt înlocuite cu documente şi informaţii. Rolul platformei colaborative este acela de a asigura instrumentele necesare comunităţilor de cercetare, în domeniul unui anumit proiect, cu scopul creşterii competitivităţii.

Obiectivele generale sunt:

- constituirea unui spaţiu virtual pentru gestiunea producţiei intelectuale din domeniul proiectului;
- construirea unei structuri de informaţii şi cunoştinţe în format electronic, clasificate pe diferite arii sau domenii de interes, astfel încât să se optimizeze timpul şi costurile de accesare a informaţiilor pertinente şi relevante, necesare diverselor activităţi din cadrul proiectului;
- realizarea unui sistem de stimulare şi tranzacţionare a cunoştinţelor în sprijinul procesului de inovare;
- gestiunea fluxurilor de informaţii şi comunicaţie între membrii consorţiului;
- gestiunea accesului rapid la resursele documentare ale consorţiului, reducerea timpilor de căutare a informaţiilor şi accelerarea proceselor de cercetare;
- identificarea contextuală a informaţiilor utile proiectului;
- evaluarea valorică a contribuţiilor membrilor consorţiului;
- documentarea membrilor echipelor de cercetare şi stimularea colaborării;

- reducerea impactului migraţiei personalului din cercetare prin retenţia contribuţiilor de valoare, respectiv accelerarea integrării în sistem a tinerilor cercetători;
- durabilitatea şi continuitatea procesului de cercetare prin menţinerea şi actualizarea permanentă a stadiului activităţii de cercetare.

Conceptul de platformă colaborativă este menit să asigure o reunire a diferitelor colective, care să formeze grupuri de lucru, în cadrul unor proiecte de cercetare-dezvoltare. Ca organizaţii interesate, se au în vedere institute naţionale de cercetare, firme reprezentative, universităţi cu centre de cercetare-dezvoltare, asociaţii profesionale, centre de excelenţă etc. Rolul de conducător al unei platforme poate fi deţinut de un centru de excelenţă, de un institut de cercetare, de o universitate, prin intermediul căreia se pot constitui grupuri de lucru, sau grupuri de reprezentare la nivel naţional, pe domenii, care să înglobeze parteneri interesaţi în desfăşurarea unor activităţi de cercetare-dezvoltare.

Etapele de formare a unei platforme colaborative sunt:

Etapa 1 : Intâlnirea părţilor interesate, pentru a stabili o viziune comună asupra realizării proiectului.

Etapa 2 : Definirea de către parteneri a unei Agende Strategice de Cercetare (ASC).

Etapa 3 : Implementarea ASC stabilită de către membrii grupului de lucru, în jurul unor proiecte.

Etapa 4 : Proiectarea platformei colaborative.

In domeniul infrastructurii, sunt implicate, în principal, resurse cum ar fi servere, medii de stocare, sisteme distribuite şi reţele. In viziunea cerinţelor actuale, arhitectura orientată pe servicii − Service Oriented Architecture (SOA) a devenit principala arhitectură aflată la baza realizării platformelor. SOA furnizeaza sisteme complexe într-o configuraţie în format modular.

Semantica este un element cheie pentru transformarea informaţiilor în cunoştinţe. O cale de a construi aceste cunoştinţe este de a găsi metode avansate care să permită cautări rapide într-un volum mare de date nestructurate. Prin tehnologiile web, există posibilitatea de a comunica de la maşina la maşina şi de a manipula date.

Realizarea unui serviciu include mai multe etape:

- *Crearea* - realizarea conţinutului primar al serviciului;
- *Combinarea* - crearea contextului care susţine construirea unui set de aplicaţii cu conţinut specific, destinat unui anumit segment de utilizatori sau unor situaţii precizate;
- *Conectarea* - realizarea condiţiilor pentru conectarea persoanelor şi a dispozitivelor, în vederea unor legături de comunicaţie între acestea;
- *Găzduirea* - stabilirea unor modalităţi pentru oferirea de suport de stocare pentru comunicaţii, operaţii, servicii, aplicaţii şi utilităţi.

## 4.3. Platforma colaborativă CALCULOS

Platforma colaborativă CALCULOS utilizează o Arhitectura Client-Server. Clientul accesează platforma utilizând un browser de pe un calculator, selectează o locaţie pentru a trimite o cerere către server, cererea ajunge la server, care răspunde şi trimite răspunsul către client.

Această arhitectură presupune:

- Existenţa unui calculator (clientul) care formulează o cerere;
- Cererea clientului este expediată unui server;
- Serverul analizează această cerere, o execută, formulează răspunsul şi îl expediază clientului;
- Clientul recepţionează răspunsul la cererea solicitată;
- Cererea clientului poate implica şi consultarea unei baze de date aflate pe un server de baze de date.

Arhitectura Client-Server, cu relaţiile sale, este prezentată în Figura 16:

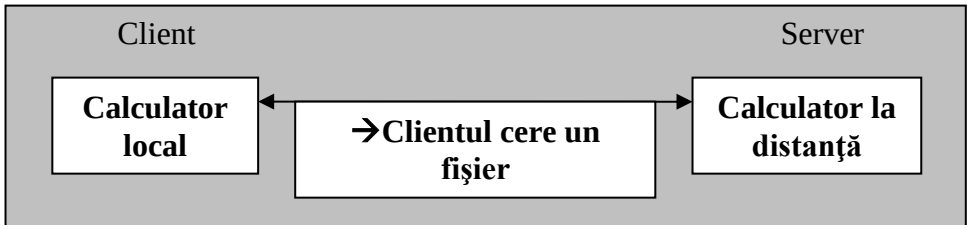



Fig. 16. Arhitectura client - server

Mai mulţi clienţi pot formula simultan cereri către acelaşi server. Serverul poate răspunde clientului :
- oferind informaţia solicitată;
- anunţând că nu este autorizat să acceseze fişierul (informaţia) respectiv(ă);
- anunţând că fişierul (informaţia) căutat(ă) nu a fost găsit(ă) pe acest server.

O aplicaţie pe trei niveluri este împărţită conform Figurii 17:

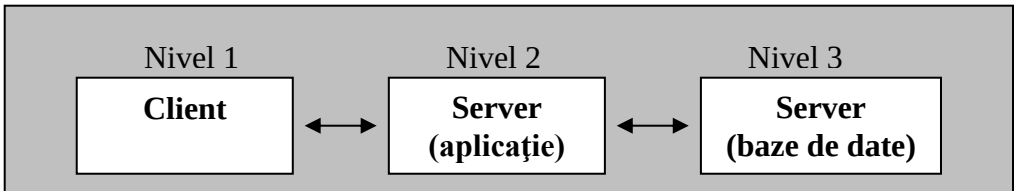



Fig. 17. Aplicaţie pe trei niveluri

Aplicaţiile pe 3 niveluri au precedat apariţia WWW (World Wide Web), dar acesta a influenţat într-un mod hotărâtor noile aplicaţii, deoarece o aplicaţie Web se bazează pe un standard deschis şi larg acceptat. Aplicaţia foloseşte un browser Web pentru rularea interfeţei cu utilizatorul. Acest lucru înseamnă că rulează într-un mediu TCP/IP şi utilizează protocolul HTTP pentru a comunica cu serverul. Protocolul HTTP foloseşte HTML şi alte tipuri de date MIME pentru datele pe care le transferă între server şi client.

Schema de proiectare pentru o platformă colaborativă împarte aplicaţia în trei mari părţi componente: Prezentare, Model şi Baza de date. Acestea reprezintă, în general, componentele logice ale unei aplicaţii web. Prezentarea (sau logica prezentării) afişează utilizatorului diferite părţi şi aspecte ale Modelului şi Bazei de date (sau logica funcţională), permite utilizatorului să modifice valorile Modelului, sau să schimbe modul în care Modelul este prezentat. De cele mai multe ori, utilizatorul nici nu observă existenţa Bazei de date. In forma sa cea mai simplă, el nu se ocupă de comunicarea dintre

celelalte două componente. In funcţie de necesităţile proiectului, se pot realiza unele variante de amplasament a aplicaţiei şi a logicii funcţionale.

In general, orice aplicaţie Web are la bază trei componente :

- baza de date;
- programele care rulează pe server şi care reprezintă modulul cel mai important al unei aplicaţii WEB;
- subsistemul "client".

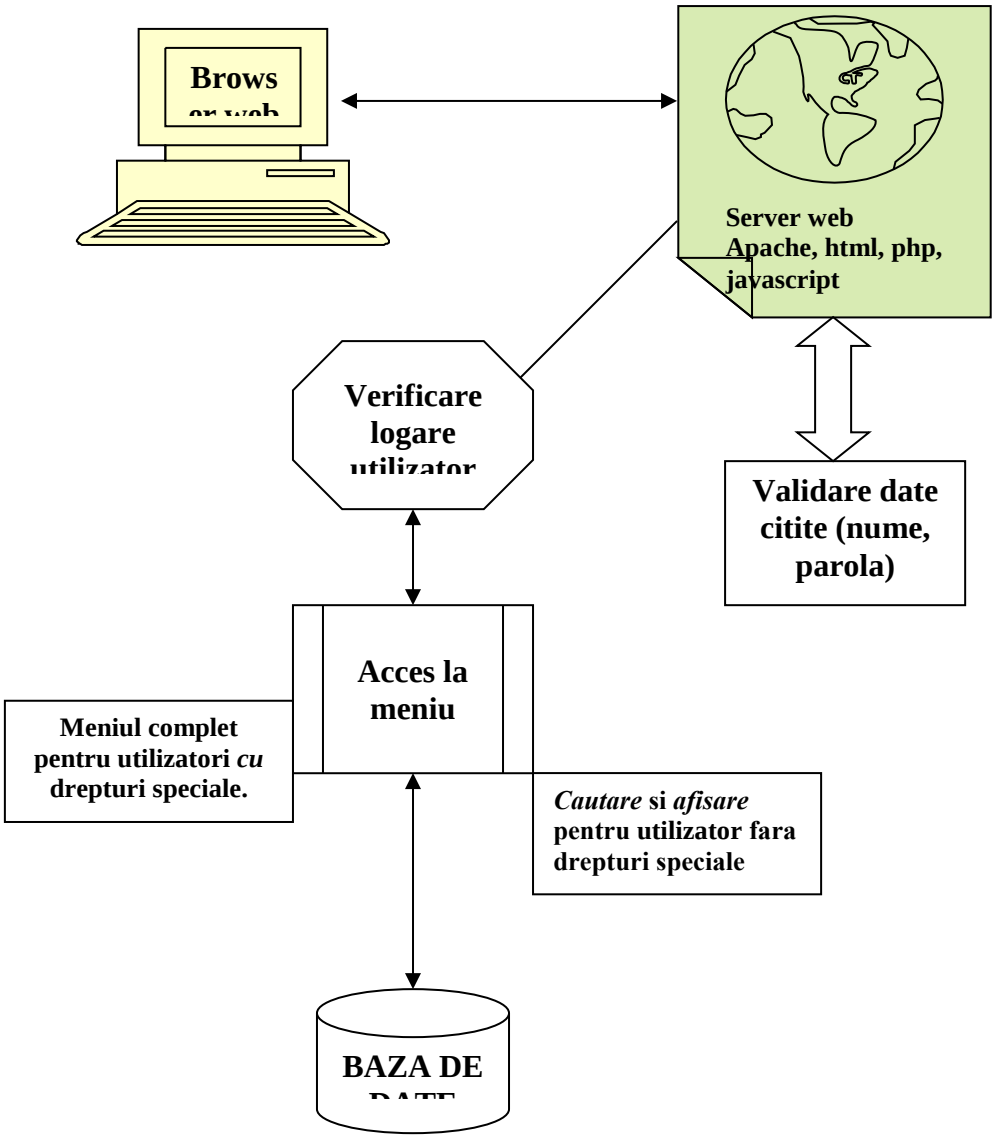



Fig. 18. Structura unei aplicaţii Web

In cazul platformei colaborative CALCULOS, baza de date a fost construită folosind tehnologia Oracle. A fost ales acest sistem de bază de date deoarece este robust, stabil, distribuit şi oferă o mare siguranţă a datelor, oricât de numeroase ar fi acestea.

Programele (partea de script) reprezintă partea cheie în cadrul unei aplicaţii Client-Server, pentru că facilitează legătura între "Client" şi baza de date cu ajutorul Serverului. Au fost alese scripturi PHP pe o platforma Windows, prin intermediul sistemului de operare Windows XP, şi serviciul Apache, care este un server de Web. Cu

ajutorul acestuia se rulează scripturile PHP care efectuează transferul de date între "Client" şi baza de date. Clientul are la dispoziţie un instrument specializat, şi anume un browser, cu ajutorul căruia se leagă la server printr-o anumită adresă. Serverul se află la distanţă şi stă în aşteptare ca anumiţi clienţi să se conecteze. După ce aceştia s-au conectat, folosind un nume de utilizator şi o parolă, programul server satisface cererile făcute de client, rulând diferitele scripturi pentru realizarea funcţiilor solicitate. Un fişier cu extensia php nu conţine numai scripturi php, ci conţine şi cod html şi cod javascript. Codurile html şi javascript sunt acele coduri care pot fi interpretate de către browser. Astfel, clientul se leagă de la distanţă la un server, la o adresă care conţine, de regulă, un fişier index.php, care este fişierul de pornire. In continuare, pentru ca cererea clientului să fie satisfacută, trebuie ca acel fişier index.php să fie executat de către serverul de Web Apache, care parcurge fişierul cu extensia php şi execută acele linii care sunt cuprinse în tagul specific.

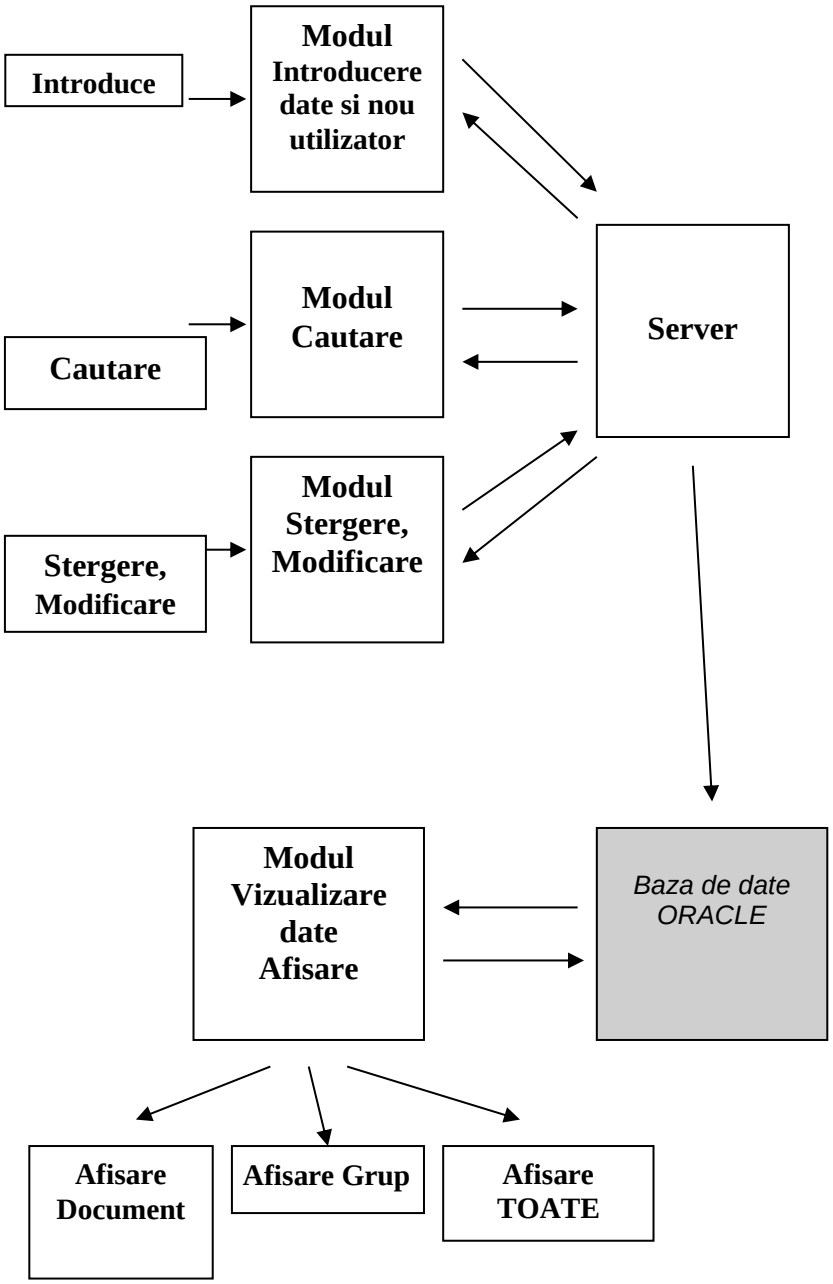
După realizarea legăturii la baza de date se pot face toate operaţiile ce ţin de baza de date, conform drepturilor acordate utilizatorului, cu ajutorul comenzilor sql. După ce aceste linii de cod php au fost executate, serverul răspunde cererii clientului. Browserul clientului primeşte acele linii ce conţin cod html şi javascript şi le interpretează, astfel încât clientul poate vizualiza rezultatul cererii. Baza de date, în care se stochează permanent informaţiile, permite introducerea de documente şi utilizatori, căutare date, afişare date, actualizare date, ştergere date.

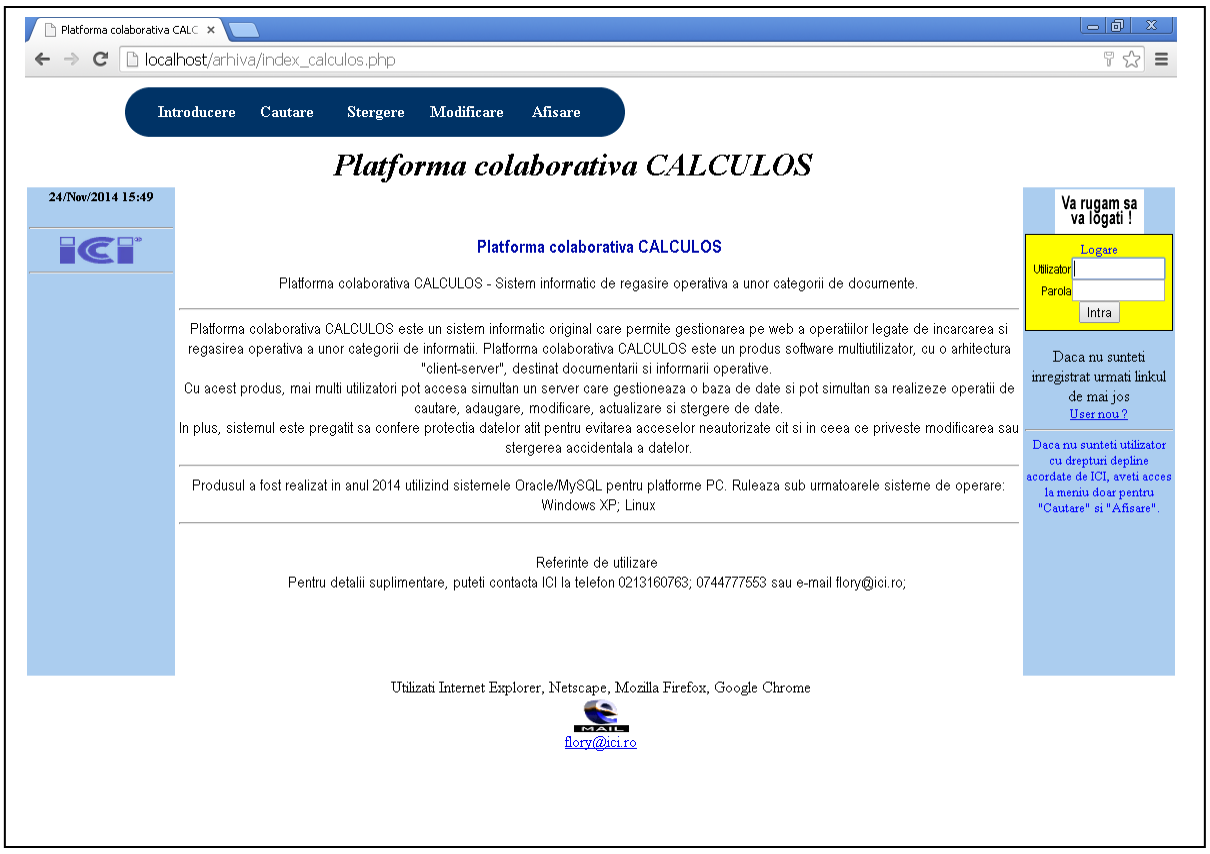
Instrumentele pentru realizarea aplicaţiilor tranzacţionale de tip client-server se bazează, în general, pe tehnici dedicate unor tipuri anume de platforme software şi hardware, implicând cerinţe specifice, referitoare la comunicarea în reţele de calculatoare, utilizarea unor anumite tipuri de baze de date şi a anumitor produse software de tip client sau server. Toate acestea conduc la realizarea unor arhitecturi mai puţin flexibile şi la elaborarea unor aplicaţii dedicate.

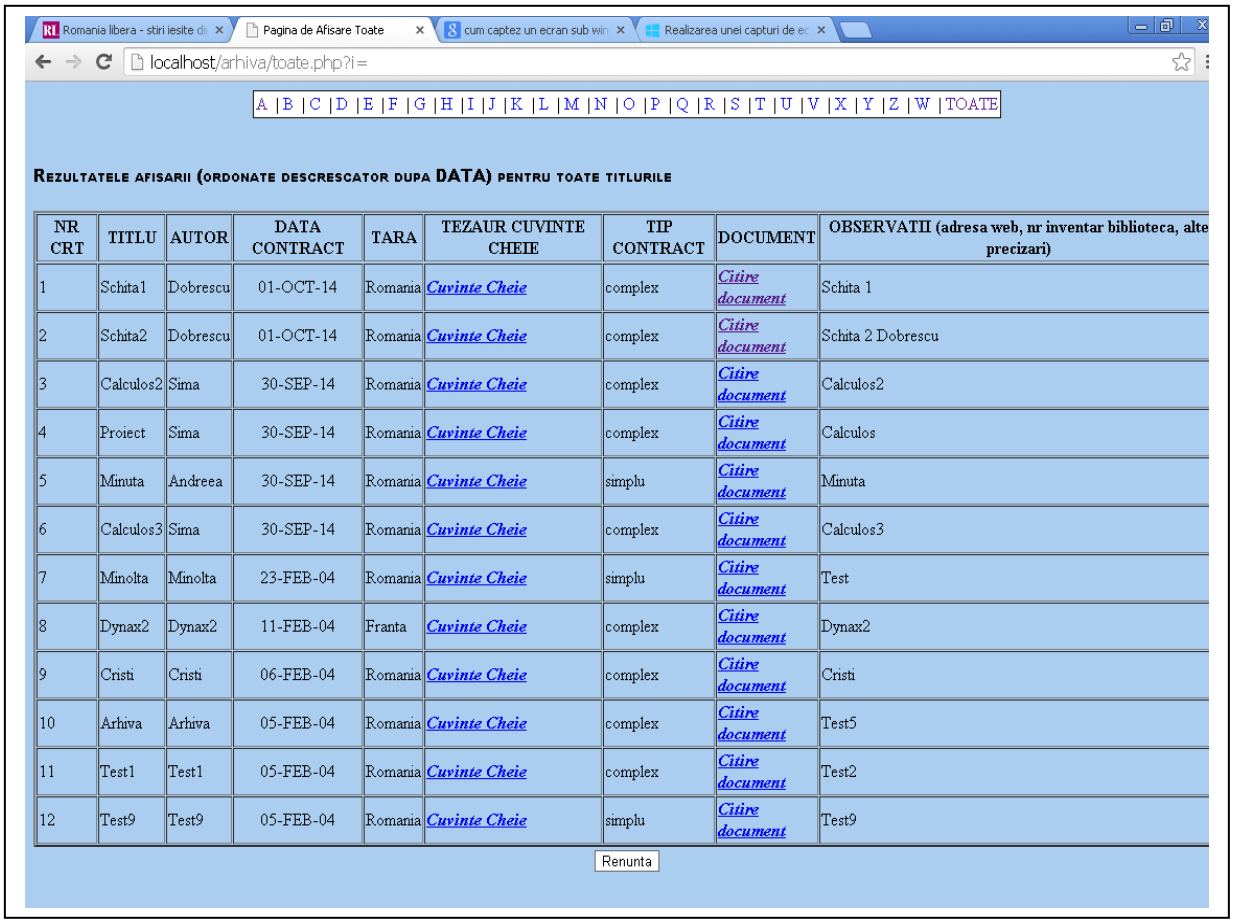
Utilizarea tehnologiilor Internet permite lucrul în reţele de calculatoare distribuite, acces global la componentele software ale sistemelor, independenţa faţă de platforma hardware şi software şi exploatarea în comun a informaţiilor. Datorită infrastructurii de comunicaţie prin Internet, este posibilă conectarea de la distanţă a participanţilor la procesul de dezvoltare a unei aplicaţii şi oferirea unui suport pentru întreg ciclul de viaţă al unui produs. Se stabileşte astfel un mediu de dezvoltare a produselor software care are o infrastructură configurabilă şi entităţi flexibile de stocare a informaţiilor. Aceste entităţi acceptă un set de instrumente software care pot coopera în cadrul unei reţele de calculatoare distribuite.

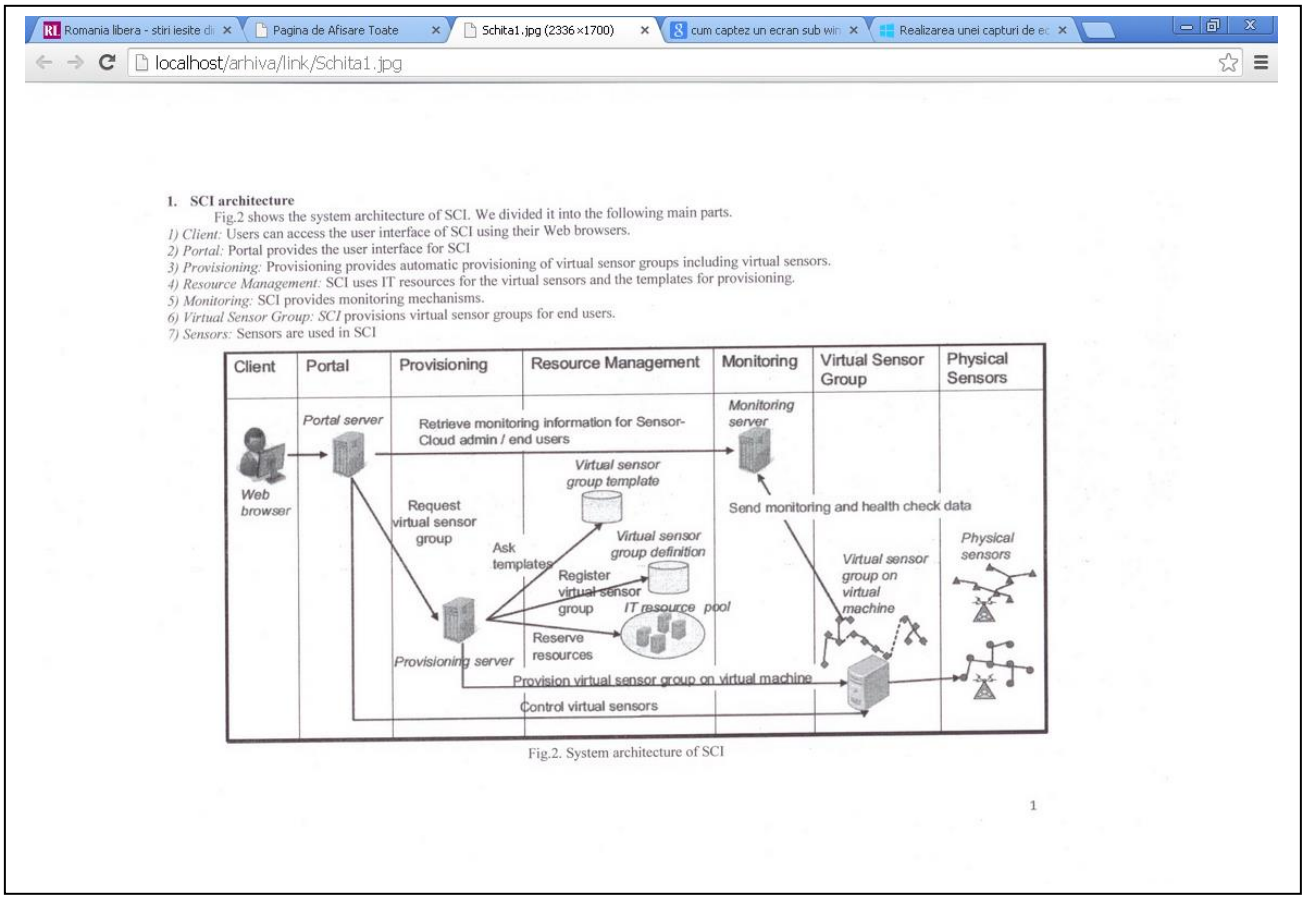
Platforma colaborativă CALCULOS cuprinde următoarele module:
- Modulul pentru introducere utilizator nou.
- Modulul pentru afişarea datelor.
- Modulul pentru ştergerea datelor.
- Modulul pentru căutarea datelor.
- Modulul pentru actualizarea datelor.

Acestea sunt ilustrate în figura următoare, iar in continuare sunt prezentate cateva ecrane ale platformei colaborative CALCULOS.


Introduce
Modul Introducere date si nou utilizator
Cautare
Modul Cautare
Server
Stergere, Modificare
Modul Stergere, Modificare
Modul Vizualizare date Afisare
Baza de date ORACLE
Afisare Document
Afisare Grup
Afisare TOATE


Platforma colaborativa CALC
localhost/arhiva/index_calculos.php
Introducere
Cautare
Stergere
Modificare
Afisare
Platforma colaborativa CALCULOS
24/Nov/2014 15:49
Platforma colaborativa CALCULOS
Platforma colaborativa CALCULOS - Sistem informatic de regasire operativa a unor categorii de documente.
Platforma colaborativa CALCULOS este un sistem informatic original care permite gestionarea pe web a operatiilor legate de incarcarea si regasirea operativa a unor categorii de informatii. Platforma colaborativa CALCULOS este un produs software multiutilizator, cu o arhitectura "client-server", destinat documentarii si informarii operative.
Cu acest produs, mai multi utilizatori pot accesa simultan un server care gestioneaza o baza de date si pot simultan sa realizeze operatii de cautare, adaugare, modificare, actualizare si stergere de date.
In plus, sistemul este pregatit sa confere protectia datelor atit pentru evitarea acceselor neautorizate cit si in ceea ce priveste modificarea sau stergerea accidentala a datelor.
Produsul a fost realizat in anul 2014 utilizind sistemele Oracle/MySQL pentru platforme PC. Ruleaza sub urmatoarele sisteme de operare: Windows XP; Linux
Referinte de utilizare
Pentru detalii suplimentare, puteti contacta ICI la telefon 0213160763; 0744777553 sau e-mail flory@ici.ro;
Va rugam sa va logati !
Logare
Utilizator
Parola
Intra
Daca nu sunteti inregistrat urmati linkul de mai jos
User nou?
Daca nu sunteti utilizator cu drepturi depline acordate de ICI, aveti acces la meniu doar pentru "Cautare" si "Afisare".
Utilizati Internet Explorer, Netscape, Mozilla Firefox, Google Chrome
MAIL
flory@ici.ro


Romania libera - stiri iesite di
Pagina de Afisare Toate
cum captez un ecran sub win
Realizarea unei capturi de ec
localhost/arhiva/toate.php?i=
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | X | Y | Z | W | TOATE
REZULTATELE AFISARII (ORDONATE DESCRESCATOR DUPA DATA) PENTRU TOATE TITLURILE
NR CRT | TITLU | AUTOR | DATA CONTRACT | TARA | TEZAUR CUVINTE CHEIE | TIP CONTRACT | DOCUMENT | OBSERVATII (adresa web, nr inventar biblioteca, alte precizari)
1 | Schita1 | Dobrescu | 01-OCT-14 | Romania | Cuvinte Cheie | complex | Citire document | Schita 1
2 | Schita2 | Dobrescu | 01-OCT-14 | Romania | Cuvinte Cheie | complex | Citire document | Schita 2 Dobrescu
3 | Calculos2 | Sima | 30-SEP-14 | Romania | Cuvinte Cheie | complex | Citire document | Calculos2
4 | Proiect | Sima | 30-SEP-14 | Romania | Cuvinte Cheie | complex | Citire document | Calculos
5 | Minuta | Andreea | 30-SEP-14 | Romania | Cuvinte Cheie | simplu | Citire document | Minuta
6 | Calculos3 | Sima | 30-SEP-14 | Romania | Cuvinte Cheie | complex | Citire document | Calculos3
7 | Minolta | Minolta | 23-FEB-04 | Romania | Cuvinte Cheie | simplu | Citire document | Test
8 | Dynax2 | Dynax2 | 11-FEB-04 | Franta | Cuvinte Cheie | complex | Citire document | Dynax2
9 | Cristi | Cristi | 06-FEB-04 | Romania | Cuvinte Cheie | complex | Citire document | Cristi
10 | Arhiva | Arhiva | 05-FEB-04 | Romania | Cuvinte Cheie | complex | Citire document | Test5
11 | Test1 | Test1 | 05-FEB-04 | Romania | Cuvinte Cheie | complex | Citire document | Test2
12 | Test9 | Test9 | 05-FEB-04 | Romania | Cuvinte Cheie | simplu | Citire document | Test9
Renunta




**1. SCI architecture**

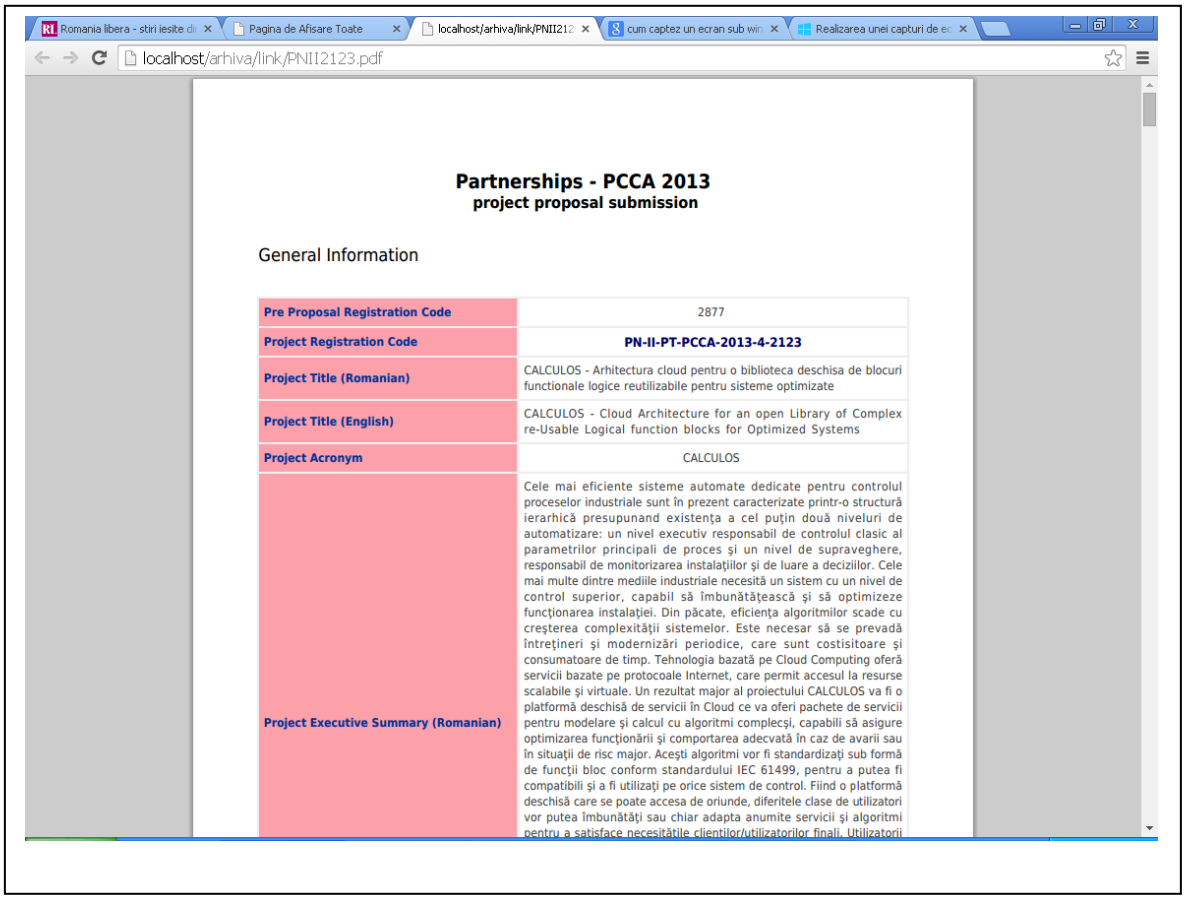
Fig.2 shows the system architecture of SCI. We divided it into the following main parts.

*1) Client:* Users can access the user interface of SCI using their Web browsers.
*2) Portal:* Portal provides the user interface for SCI
*3) Provisioning:* Provisioning provides automatic provisioning of virtual sensor groups including virtual sensors.
*4) Resource Management:* SCI uses IT resources for the virtual sensors and the templates for provisioning.
*5) Monitoring:* SCI provides monitoring mechanisms.
*6) Virtual Sensor Group: SCI* provisions virtual sensor groups for end users.
*7) Sensors:* Sensors are used in SCI



Fig.2. System architecture of SCI

1




**Partnerships - PCCA 2013**
**project proposal submission**

General Information

| | |
|---|---|
| **Pre Proposal Registration Code** | 2877 |
| **Project Registration Code** | **PN-II-PT-PCCA-2013-4-2123** |
| **Project Title (Romanian)** | CALCULOS - Arhitectura cloud pentru o biblioteca deschisa de blocuri functionale logice reutilizabile pentru sisteme optimizate |
| **Project Title (English)** | CALCULOS - Cloud Architecture for an open Library of Complex re-Usable Logical function blocks for Optimized Systems |
| **Project Acronym** | CALCULOS |
| **Project Executive Summary (Romanian)** | Cele mai eficiente sisteme automate dedicate pentru controlul proceselor industriale sunt în prezent caracterizate printr-o structură ierarhică presupunand existența a cel puțin două niveluri de automatizare: un nivel executiv responsabil de controlul clasic al parametrilor principali de proces și un nivel de supraveghere, responsabil de monitorizarea instalațiilor și de luare a deciziilor. Cele mai multe dintre mediile industriale necesită un sistem cu un nivel de control superior, capabil să îmbunătățească și să optimizeze funcționarea instalației. Din păcate, eficiența algoritmilor scade cu creșterea complexității sistemelor. Este necesar să se prevadă întrețineri și modernizări periodice, care sunt costisitoare și consumatoare de timp. Tehnologia bazată pe Cloud Computing oferă servicii bazate pe protocoale Internet, care permit accesul la resurse scalabile și virtuale. Un rezultat major al proiectului CALCULOS va fi o platformă deschisă de servicii în Cloud ce va oferi pachete de servicii pentru modelare și calcul cu algoritmi complecși, capabili să asigure optimizarea funcționării și comportarea adecvată în caz de avarii sau în situații de risc major. Acești algoritmi vor fi standardizați sub formă de funcții bloc conform standardului IEC 61499, pentru a putea fi compatibili și a fi utilizați pe orice sistem de control. Fiind o platformă deschisă care se poate accesa de oriunde, diferitele clase de utilizatori vor putea îmbunătăți sau chiar adapta anumite servicii și algoritmi pentru a satisface necesitățile clienților/utilizatorilor finali. Utilizatorii |

Figura 19 prezintă structura platformei colaborative (CP). Aceasta va fi utilizată pentru a asigura schimbul de informaţii între partenerii de proiect şi utilizatorii înregistraţi care doresc să contribuie la analiza specificaţiilor utilizatorilor şi să testeze rezultatele proiectului chiar din faza de dezvoltare. Utilizatorii vor fi încurajaţi să descarce funcţiile bloc şi să le testeze pentru diverse instalaţii din diferite domenii industriale, utilizând infrastructura de modelare bazată pe tehnologii cloud, ce suportă simulare în buclă cu modele de instalaţii. Pe baza răspunsului primit de la utilizatorii finali, se va efectua un proces de selectare, astfel încât numai cei mai eficienţi algoritmi vor fi păstraţi în biblioteca bazei de date, în timp ce acei algoritmi mai neperformanţi vor fi eliminaţi sau optimizaţi, pentru a îndeplini criteriile utilizatorilor. Aceste date vor fi stocate utilizând etichetele semantice corespunzatoare pentru căutari ulterioare mai eficiente.

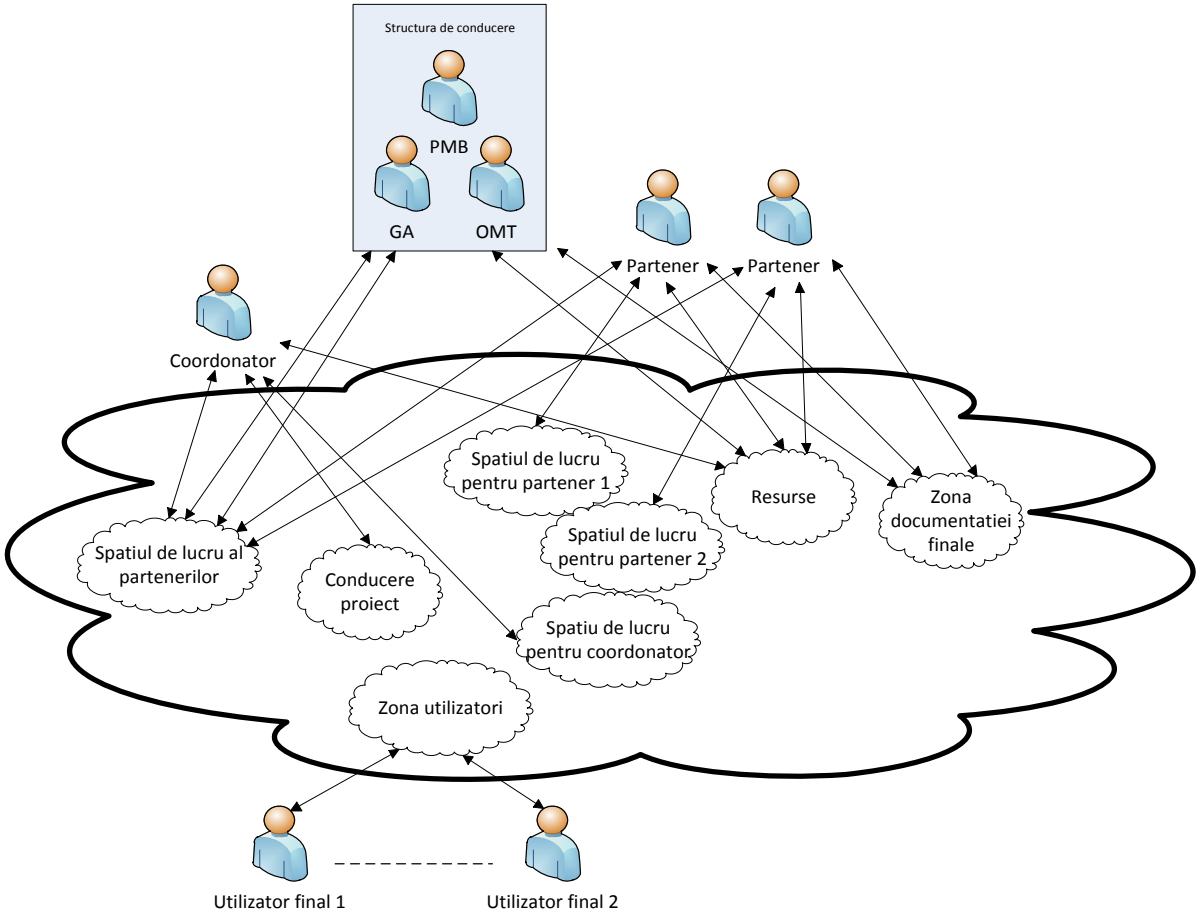



Fig. 19. Platforma Colaborativă

Diferitele categorii de utilizatori vor putea accesa această platformă colaborativă în diverse scopuri, în funcţie de statutul lor. Astfel, Administratorul Bazei de Date va fi responsabil de monitorizarea drepturilor şi statuturilor utilizatorilor şi va asigura suportul tehnic şi mentenanţa pe acest server. Când un utilizator final/client furnizează date despre proces în vederea creării unui algoritm de control pentru o anumită instalaţie, informaţiile vor fi strict confidenţiale. Inginerii specializaţi pot coopera în diverse faze ale procesului de elaborare a unui algoritm, pentru a identifica cele mai bune soluţii de implementare a funcţionalităţii acestuia. Ei colaborează pentru a identifica cele mai

bune soluţii pentru client, dar vor pune la dispoziţie şi serviciul de mentenanţă pentru modulul în care sunt specializaţi. Ei au acces numai la anumite module specifice şi la modulele de validare, simulare şi testare. Astfel, CALCULOS ajută la îmbunătăţirea relaţiei dintre mediul academic şi companiile industriale. Un alt tip de utilizator este potenţialul client. Asemenea utilizatori au acces la modulul de bază de date, în care ei pot vedea algoritmii, îi pot testa şi primi răspuns/reacţii. Interfaţa cu utilizatorul ghidează clienţii şi îi informează pe clientii potentiali despre ofertele de suport tehnic. Proiectul va trata şi problematica defectoscopiei şi diagnozei, precum şi cea a prevenirii situaţiilor periculoase, aspecte de importanţă şi relevanţă capitală pentru multe procese industriale.

Platforma Colaborativă este susţinută de către Platforma Cloud, după cum este ilustrat în Fig. 20. Prin Platforma Colaborativă este creată o interfaţă între utilizatori şi serviciile cloud. Toate celelalte părţi implicate sunt clasificate în diferite tipuri de utilizatori.

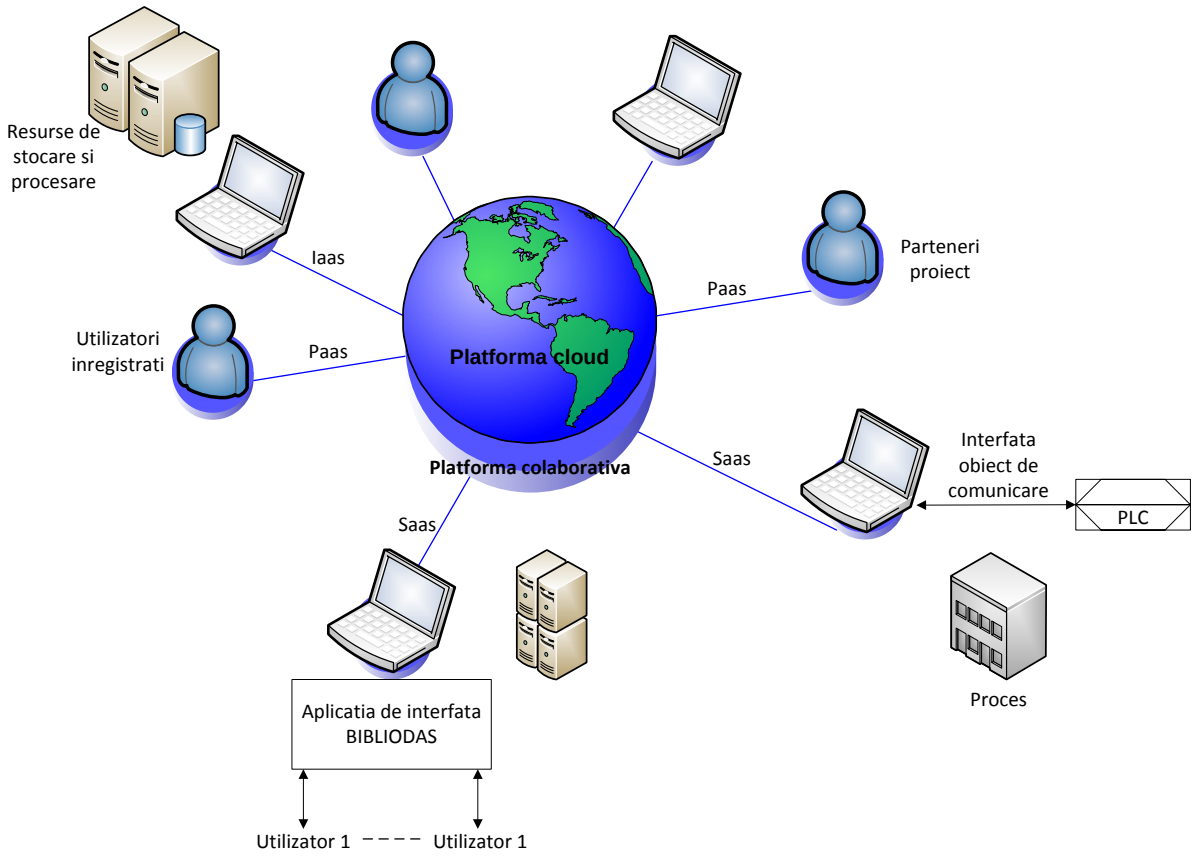



Fig. 20. Integrarea platformei colaborative într-o structură de tip cloud

Fluxul informaţiilor în sistem foloseşte o structură ierarhizată ilustrată în Fig. 21. Modulul de Baze de Date, amplasat la nivelul 0, serveşte la stocarea informaţiilor încărcate de către utilizatori prin API. Acest modul va prezenta o aplicaţie de data-mining, capabilă să facă un prim pas în descoperirea informaţiilor în baza de date (KDD– Knowledge Discovery in Databases) şi pre-procesarea informaţiilor. De asemenea, platforma asigură un Modul de Maşină Virtuală, în care

utilizatorii privilegiaţi interesaţi pot avea acces la baza de date nu numai pentru a vizualiza algoritmii, dar şi pentru a-i testa şi a oferi reacţii, sau, în cazul unor utilizatori avansaţi, chiar pentru a contribui la extinderea bazei de date. Acest lucru va servi la exploatarea rapidă a rezultatelor şi, potrivit necesităţii pentru un algoritm specific în procesul industrial, la obţinerea unei valori adăugate mari prin implementarea sa. Mai mult, având un punct valid de plecare pentru rezolvarea problemelor specifice, abordarea va conduce la obţinerea unor produse mai fiabile, îmbunătăţite, într-o gamă largă de sisteme de control al proceselor (de la producţie discretă până la instalaţii industriale cu funcţionare continuă).

Modulul de Testare şi Validare, localizat la nivelul 1, este capabil să verifice informaţia cuprinsă, colectată de la nivelul inferior şi să trimită un răspuns. Când informaţiile sunt pre-procesate, atunci acestea sunt trimise la nivelul 2, Modulul de Modelare a Proceselor, unde se creează structura procesului pe baza informaţiilor procesate la nivelul inferior. Se poate folosi un Modul de Identificare Parametrică, amplasat la acel nivel, pentru a obţine coeficienţii modelului, pe baza informaţiilor de intrare/ieşire şi a altor informaţii încărcate de către utilizatori.

Structura sistemului de control include Modulul de Dezvoltare a Algoritmilor şi Modulul de Simulare; rezultatele sunt accesibile utilizatorilor prin interfaţa aplicaţiei. Modelarea şi simularea sunt necesare pentru configurarea unui algoritm pentru o anumită instalaţie, respectiv, verificarea aplicabilităţii algoritmului pentru un sistem specific, iar apoi răspunsul obţinut din proces va fi folosit în optimizarea algoritmului. Nivelul superior este reprezentat prin Modulul de Dezvoltare a Funcţiilor bloc.

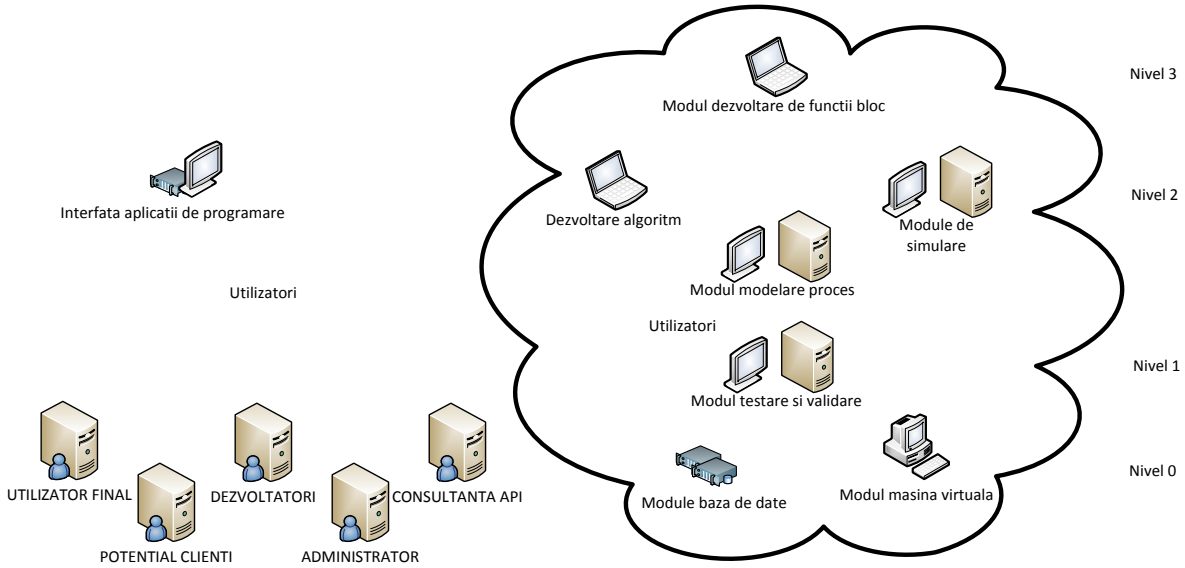



Fig. 21. Interfaţa Aplicaţiei de Programare

## 4.4. Infrastructura de preluare în cloud a datelor de la senzori fizici

**Scop**. Se propune o infrastructură dedicată, denumită Sensor-Cloud Infrastructure (SCI), care permite virtualizarea unui senzor fizic pentru acces în cloud și, prin extensie, virtualizarea unei rețele de senzori fizici, care devine astfel parte componentă în cloud. În particular, a fost tratată

doar problema virtualizării reţelelor de senzori wireless (WSN). Interacţiunea dintre cloud şi lumea reală, reprezentată de WSN, este decuplată, în sensul că toate operaţiile executate în cloud se fac cu date furnizate de senzorul virtual). Ca atare, interacţiunile pot fi grupate în două categorii: cloud vs. senzor virtual și, respectiv, senzor virtual vs. nod din reţeaua de senzori reali.

**Avantajele solutiei propuse**:
- Nu mai este necesară instalarea de sisteme SCADA pe calculatoarele locale;
- Utilizatorii au acces la diverse tehnologii informatice cu suport off-side;
- Se oferă un spaţiu scalabil de server într-un cloud privat;
- Pot fi satisfăcute la parametri de calitate superioară cerinţele de timp real.

**Arhitectura SCI**. Arhitectura software SCI are şapte componente (vezi fig.9).
*1) Client:* Utilizatorii pot accesa interfaţa utilizator a SCI prin browsere Web.
*2) Portal:* Portalul furnizează interfaţa utilizator a SCI.
*3) Alocare* (*provisioning*)*:* Punere în funcţiune/alocare automată a senzorilor virtuali.
*4) Managementul resurselor:* SCI utilizează resurse IT pentru senzorii virtuali.
*5) Monitorizare:* SCI furnizează mecanisme dedicate de supraveghere și monitorizare.
*6) Gruparea Senzorilor Virtuali:* SCI permite gruparea senzorilor la utilizatorii finali.
*7) Senzori:* Senzori fizici (reali) utilizaţi în SCI.

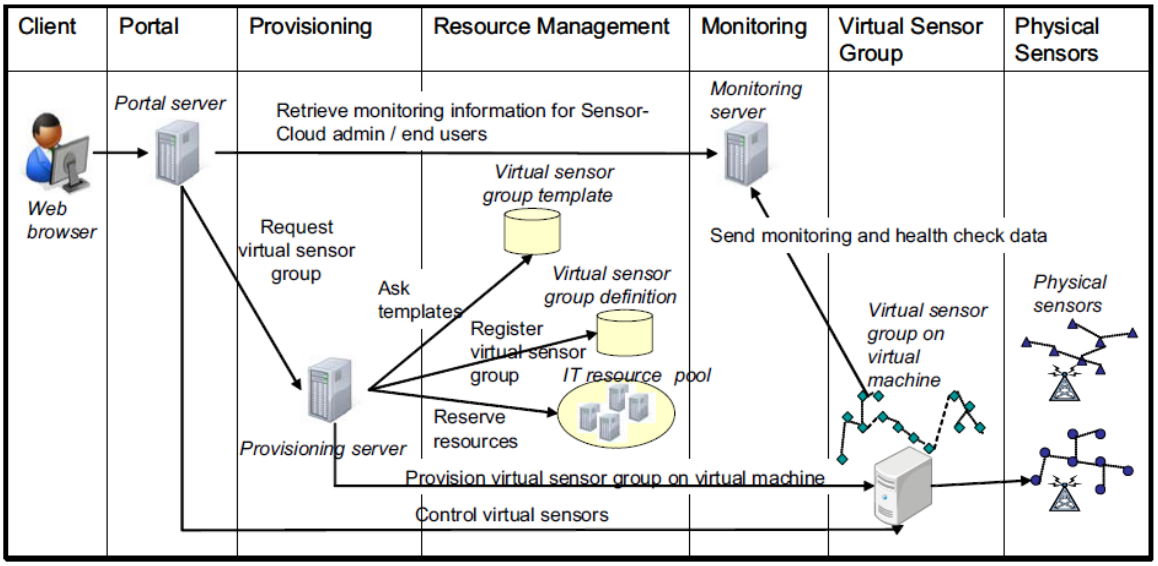


Fig.22. Arhitectura unui sistem SCI

*Functiile componentelor*
*1) Server Portal:* Atunci când un utilizator se conectează la portal printr-un browser Web, trebuie să-și precizeze statutul (utilizator final, proprietar al reţelei de senzori sau administrator SCI), pentru a determina operaţiile admise. Pentru utilizatorii finali, serverul portal indică meniurile de conectare/deconectare, cererile de alocare sau de anulare a grupurilor de senzori virtuali, precum și procedurile de monitorizare și control ale acestora. Pentru proprietarii reţelelor de senzori, serverul portal indică meniurile de conectare/deconectare și de înregistrare sau ștergere a senzorilor fizici.

*2) Serverul de alocare (Provisioning Server).* Acest server transmite grupurilor de senzori virtuali cererile de alocare de la serverul portal. Totodată, el definește fluxurile de lucru pe care le execută în ordinea prestabilită.
*3) Grup de senzori virtuali.* Un grup de senzori virtuali este alocat automat unui server virtual de serverul de alocare. El poate fi activat sau dezactivitat de proprietarul grupului și poate fi controlat de acesta direct sau printr-un browser Web.
*4) Serverul de Monitorizare* primește date despre senzorii virtuali de la agenții din serverele virtuale, date care sunt preluate de administratorul SCI.

*Fluxul de activitati pentru alocarea grupurilor de senzori virtuali*
1) Conectare.
2) Selectarea de formulare (*templates)* pentru grupurile de senzori virtuali.
3) Solicitarea unui grup de senzori virtuali prin selectarea formularelor adecvate prin portal. Portalul apelează la rândul său serverul de alocare.
4) Rezervarea resurselor IT.
5) Alocarea grupului de senzori virtuali pe serverul virtual selectat.
6) Notificarea reușitei alocării.

**Detalii de implementare**. Fig. 22 ilustrează schema de implementare.

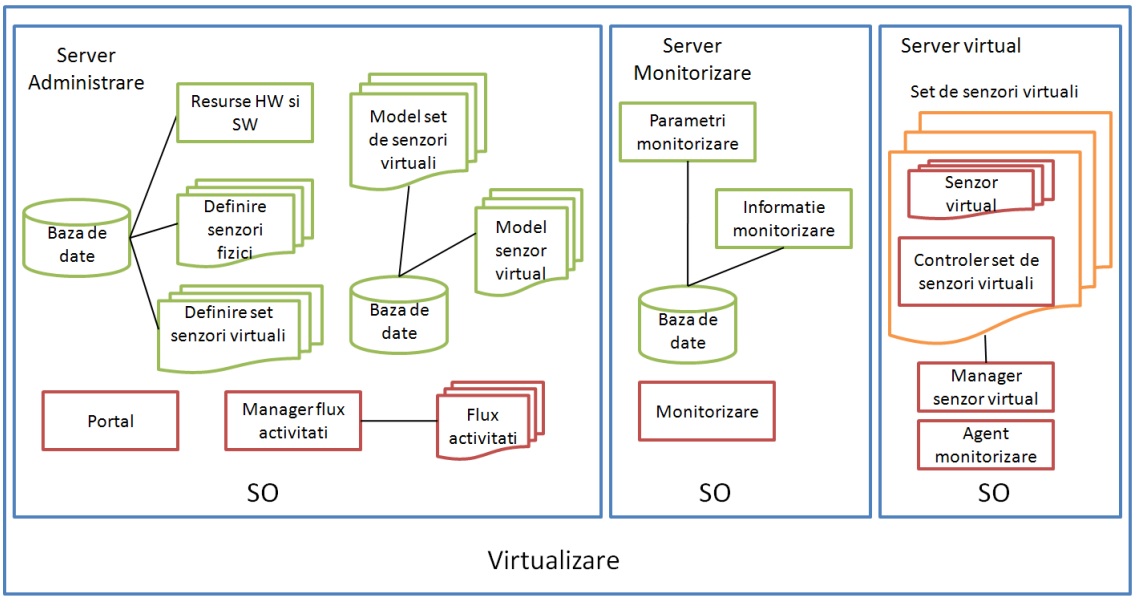



Fig.22. Prototip SCI

In această variantă, serverul de administrare asigură funcţiile atât pentru serverul portal cât și pentru cel de alocare, asigurând și managementul bazei de date. Baza de date stochează caracteristicile senzorilor fizici (ID-ul proprietarului, tipul de senzor și sursa datelor preluate de senzor), ale grupurilor de senzori virtuali (ID-urile grupului de senzori virtuali, ale utilizatorului final și, respectiv, serverului virtual, cât și datele create) și ale resurselor IT (date despre servere și despre servere virtuale, ca de exemplu, adresa IP sau numele gazdei). Totodată, se crează un registru depozit care stochează formularele tipizate pentru senzorii virtuali (un astfel de formular conţine biblioteca de programe și fișierele cu reguli de prelucrare și clasificare a datelor).

**Descrierea fluxului de activităţi**. Fig. 23 prezintă fluxul activităţilor de alocare a unui grup de senzori virtuali.

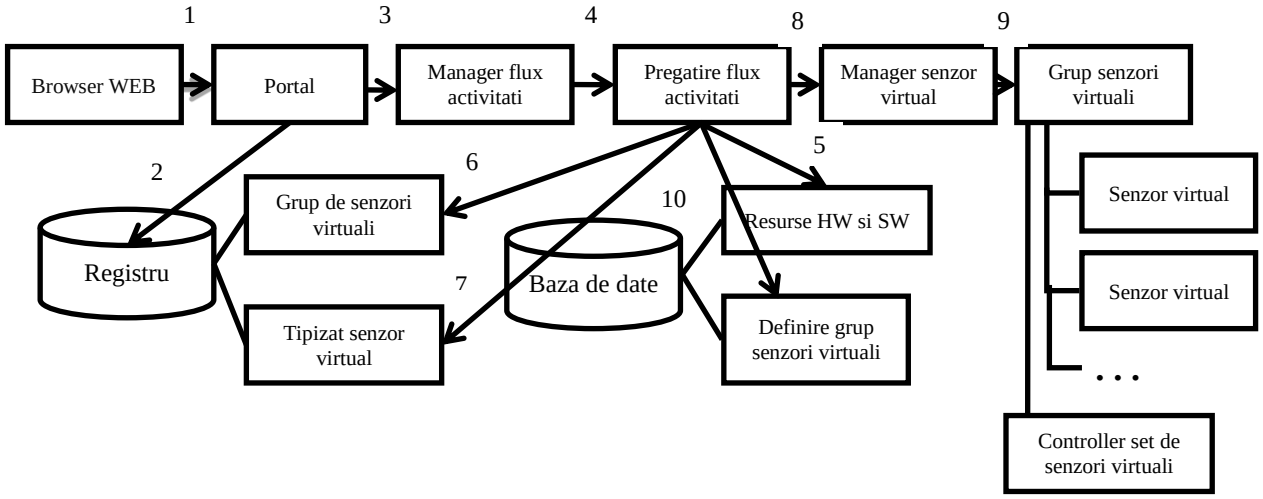



Fig. 23. Fluxul activitatilor de alocare

*1. Cerere.* Un utilizator se conectează la portal prin browser (1). Serverul portal obţine lista de formulare tip ale senzorilor virtuali și ale grupurilor de senzori virtuali de la registrul depozit (2) și o prezintă utilizatorului. Acesta solicită alocarea unui anumit grup de senzori. Serverul portal apelează managerul fluxului de activităţi cu ID-ul utilizatorului și cel al grupului de senzori virtuali prin nume (3).
*2) Apelarea managerului de activităţi.* Acest mecanism execută fluxul de activităţi de alocare a grupului de senzori virtuali (4), configurează resursele IT pentru rezervarea și furnizarea informaţiei de la grupul de senzori virtuali (5), obţine ID-ul grupului de senzori virtuali (6) și formularul tipizat al grupului de senzori virtuali (7). Totodată, instanţiază managerul senzorului virtual (8), diferiţii senzori virtuali creaţi și controllerul grupului de senzori virtuali (9).
*3) Post alocare.* La final, definiţia noului grup de senzori virtuali este înscrisă în baza de date (10), iar utilizatorul final este anunţat că alocarea a fost reușită.

# 5. Analiza posibilităţilor de interfaţare

## 5.1. Interfaţarea cu sisteme SCADA

O mare contribuție la succesul arhitecturii prezentate este adusă de modul în care este executată interfațarea. Așa cum s-a menționat anterior, pentru structurarea algoritmilor de control se va folosi conceptul de funcții bloc distribuite. Similar cu circuitele integrate utilizate în proiectarea circuitelor electronice, o funcție bloc încorporează o anumită funcționalitate și poate fi conectată la alte funcții bloc prin intrările și ieșirile sale.

In dezvoltarea algoritmilor se poate utiliza Standardul IEC 61499, ce definește o arhitectură deschisă pentru noile generații de control distribuit și automatizări. La elaborarea acestui standard, IEC a luat în considerare proprietățile de portabilitate, reutilizare, interoperabilitate și reconfigurare a aplicațiilor distribuite. Spre deosebire de predecesorul său, IEC 61131-3, o funcție bloc în IEC

61499 rămâne pasivă până în momentul în care apare un eveniment ce armează funcționalitatea respectivă și este executată, producând evenimente de ieșire și date. Dacă inițial această interfață a evenimentelor a fost criticată pentru că făcea scrierea aplicațiilor mai complicată comparativ cu IEC 61131, faptul că se permite specificarea explicită a secvenței de execuție a funcțiilor bloc dă dezvoltatorilor un nou nivel de flexibilitate inexistent anterior.

Sistemele de control industrial de tip SCADA (Supervisory Control and Data Acquisition) se află în centrul majorității industriilor moderne, cum ar fi cele de producție, energie electrică, rețele de alimentare cu apă și transport etc. Practic, în orice domeniu actual se vor găsi versiuni de sisteme SCADA, acestea implicând diverse tehnologii ce permit organizațiilor atât funcții de comandă cât și de monitorizare, colectare și prelucrare a datelor extrase din procesele industriale. Aceste sisteme variază de la configurații simple la proiecte de amploare, majoritatea acestora utilizând software de tip HMI (Human-Machine Interface) ce permite utilizatorilor să interacționeze și să controleze dispozitivele implicate (valve, pompe, motoare etc.).

SCADA primește informațiile de la RTU-uri (Remote Terminal Units) sau PLC-uri, care la rândul lor sunt alimentate cu informații de către senzori sau cu valori introduse manual. Datele colectate sunt apoi monitorizate și prelucrate cu scopul principal de a crește eficienţa și eficacitatea instalațiilor. Majoritatea sistemelor moderne SCADA permit accesarea datelor în timp real și de la mare distanță permițând luarea deciziilor imediate. Utilizarea de către programele SCADA a standardelor și practicilor IT de ultima generație, precum bazele de date puternice și integrarea cu sisteme de tip MES și ERP, permite, pe lângă o creștere a eficienţei, securității și productivității, și un flux mai facil al datelor.

Sistemele SCADA au evoluat în timp distingându-se trei generații, o arhitectură de bază fiind prezentată în Fig 24. În soluția de față se dorește ca platforma de Cloud Computing să preia atribuțiile sistemelor SCADA, adoptând tehnologia numită IoT (Internet of Things). Ca rezultat, sistemul SCADA poate raporta starea aproape în timp real și se poate folosi de proprietatea scalabilității orizontale, pe care mediul Cloud Computing o pune la dispoziție, pentru a implementa algoritmi de control mai complecși decât s-ar putea implementa în mod tradițional în PLC. Folosirea protocoalelor deschise de rețea ca TLS (intrinsec IoT) oferă în mod implicit o arie de securitate mai mare decât în cazul utilizării unui mix de protocoale de rețea proprietare, cum se întâmplă în cazul multor implementări de sisteme SCADA descentralizate. O analiză a performanţei unui sistem SCADA distribuit în cloud este efectuată în [1], implementarea utilizând patru mașini virtuale. Analiza este realizată atât din punctul de vedere al performanţei, cât și al securității. Se distinge un tip particular de cloud, anume cloud hibrid, ce este capabil să satisfacă cerințele de mentenanță, acces la distanță și management al datelor necesare unui sistem SCADA.

Cea mai mare provocare rămâne totuși cea a asigurării securității, ținând seama că unele tipuri de breșe sunt încă mai pregnante în mediul distribuit cloud, cum ar fi partajarea resurselor și imposibilitatea păstrării datelor senzitive sub controlul direct al organizației. Partajarea resurselor în cloud este realizată prin tehnologii de virtualizare, care poate fi de tip Full-Virtualization sau Para-Virtualization. În primul caz, mediul virtual este situat direct peste mediul hardware, pe când în cazul al doilea, mediul virtual este implementat indirect prin așa numitele middleware (hipervizoare), situate între sistemul de operare și hardware. O soluție care preîntâmpină unele considerente în materie de securitate asupra serviciilor oferite de furnizori este adoptarea de cloud privat și utilizarea canalelor VPN (Virtual Private Networks).

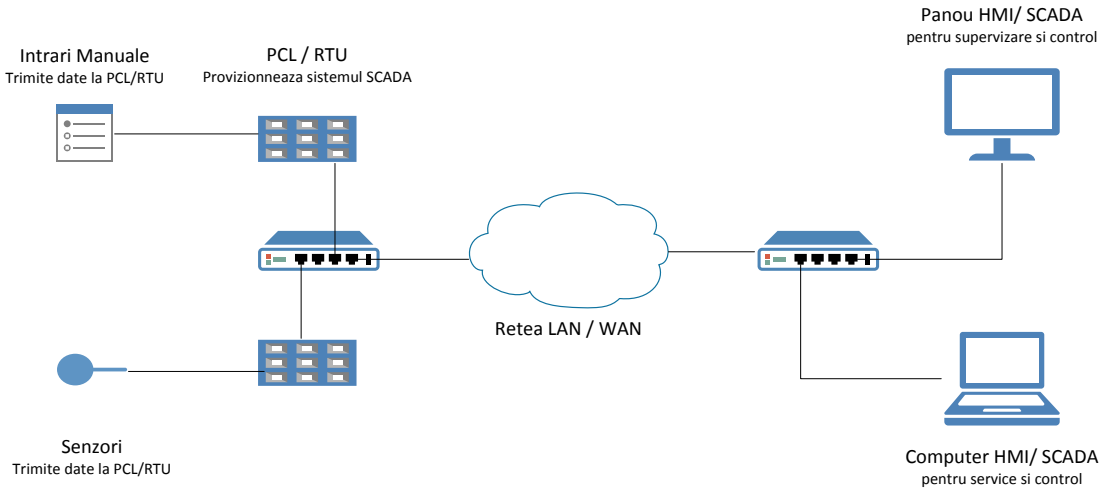



Fig. 24. Integrarea arhitecturii SCADA de bază

Fiind încă în stadiu incipient pe plan mondial, adoptarea de sisteme SCADA "în cloud" este privită rezervat, paradigma abstractizării complete a arhitecturii fizice și transferul responsabilității dezvoltatorului și utilizatorilor doar către furnizorul de servicii fiind posibilă după o periodă inițială de adaptare și probare. Se propune ca pentru sistemele SCADA complexe existente trecerea să se facă treptat, păstrând o parte din funcționalitățile esențiale în sisteme de rezervă.

## 5.2. Interfaţarea M2M pentru integrare în cloud

In accepțiunea IoT sunt interconectate trei tehnologii uzuale, senzori wireless, Internet și cloud computing, pentru a crea comunicația de tip M2M (machine-to-machine). Comunicația M2M este făcută posibilă de către senzori ce captează anumite evenimente, care sunt apoi transmise către aplicațiile ce le vor procesa. Senzorii inteligenți sunt încorporați în dispozitive aflate la distanță și transmit parametrii, precum temperatura, locația, viteza, gradul de luminozitate, parametrii medicali, etc. Senzorii includ adaptoare de comunicație wireless, capabile să primească și să transmită date prin rețelele de comunicații către un server central sau distribuit, unde aplicația le traduce în informație semnificativă ce poate fi și prelucrată. Comunicația M2M poate fi utilizată pentru aplicații din domenii variate:

- Producție: Senzorii monitorizează instalațiile de producție;
- Transport și Logistică: Urmărirea în timp real a vehiculelor, furnizând informaţii precum viteza, distanţa parcursă, consum de combustibil;
- Energie și Utilităţi: Contoare „smart" ce transmit consumul înregistrat în interval de câteva secunde;
- Servicii Publice: Orașe inteligente în care luminile stradale și semafoarele ajută la ghidarea vehiculelor ce intervin în caz de urgenţă;
- Medical: Echipament medical ce poate monitoriza pacienţii la distanţă, etc.

Dispozitivele M2M generează tipare diferite de trafic de la caz la caz, ce pot include tipare de periodicitate, bazate pe eveniment, fluxuri multimedia etc. Acest fapt rezultă de cele mai multe ori în seturi de date mari și complexe (Big Data), greu de prelucrat într-un sistem nedistribuit, cum este

cel de tip cloud. Figura 25 ilustrează amestecul de tipare de trafic prezent în comunicațiile M2M aparținând diferitelor aplicații.

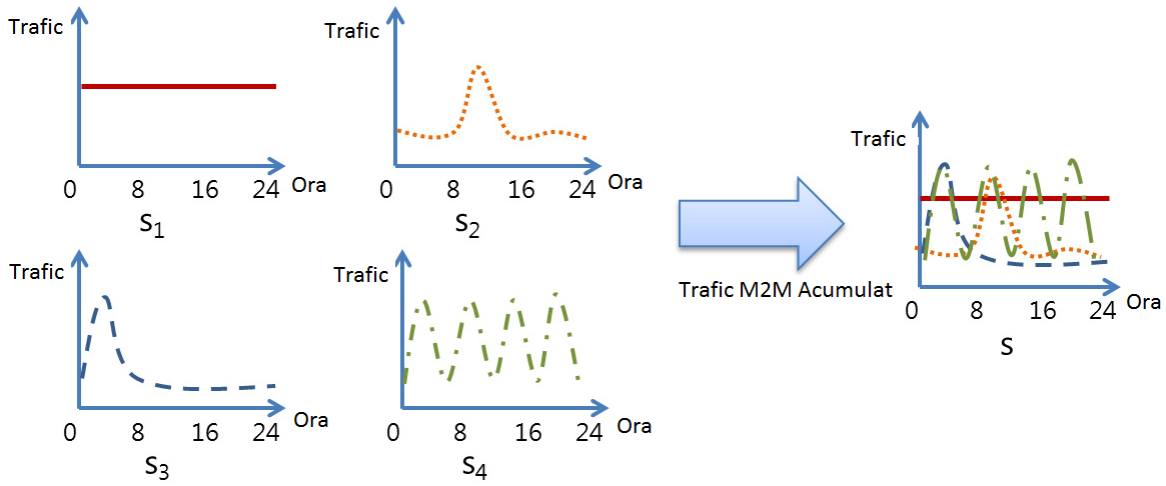



Fig. 25. Exemplu de tipar de trafic în comunicația M2M

Semnalele S1 până la S4 sunt exemple de tipare constante, bazate pe eveniment și, respectiv, periodice, generate de senzori aparținând, de exemplu, aplicațiilor de supraveghere video, urgențe, contorizare. Trebuie avut în vedere și faptul că acest trafic trebuie preluat și manevrat de mediul de comunicație. Prin mecanismele puse la dispoziție, rețelele de telecomunicații mobile pot asigura nivelul de QoS (Quality of Service) dorit.

O implementare modernă cu securitate sporită presupune că schimbul de date se va face într-un mediu controlat. Acest lucru se poate realiza încorporând senzorilor din sistem carduri SIM, efectuând astfel comunicația între senzori și platforma cloud prin intermediul unei rețele de telecomunicații mobile. Furnizorul de servicii de comunicație poate fi în același timp și cel care pune la dispoziție serviciile cloud sau doar va facilita o interconectare sigură cu acestea.

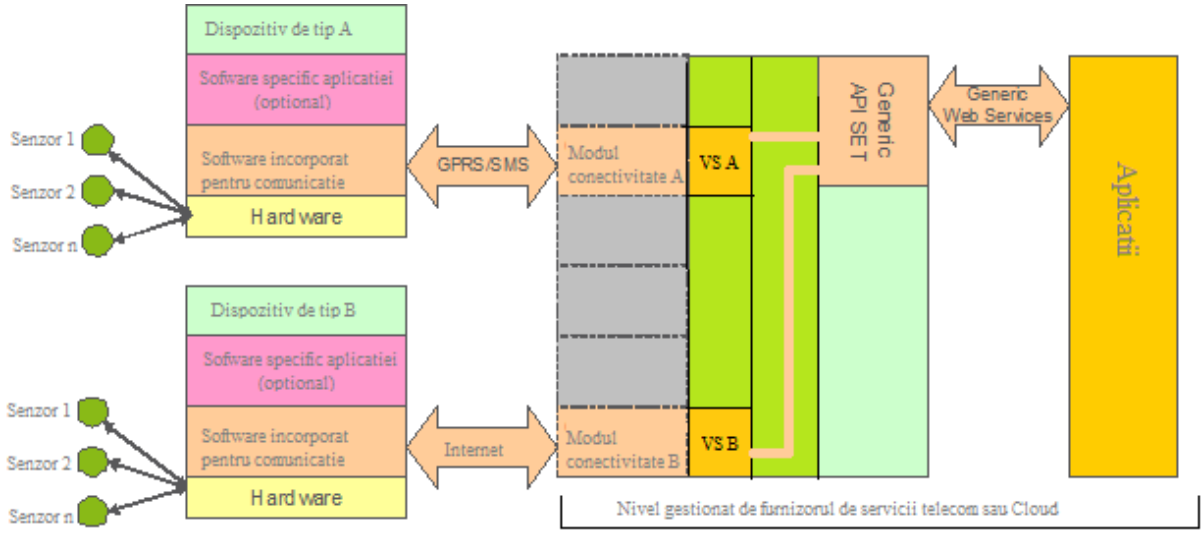



Fig. 26. Comunicație M2M între senzorii wireless și o aplicație din cloud

Figura 26 rezumă cele două variante posibile de implementare a comunicației M2M. Se propune o structură cu doi senzori virtuali (VS A, VS B), prezentă în platforma cloud, care vor putea fi controlați prin funcții standard. Alocarea și gruparea este făcută dinamic, ca răspuns la cerințele utilizatorilor sau ale aplicației. Pentru cazurile în care senzorii sunt localizați în zone ce nu pot fi acoperite în mod normal de către rețeaua radio a furnizorului de servicii de comunicații mobile (de pildă, ecranare sau interferență cu aparatura existentă), se poate adopta soluția amplasării dispozitivelor ad-hoc, numite femtocell. Acestea pot opera în locația și frecvenţa adecvată pentru a maximiza capacitatea de comunicare.

In [24] este făcută o analiză a principalelor dificultăți întâlnite la realizarea comunicației M2M a unei rețele de senzori și este prezentat un exemplu de Arhitectură de Rețea Orientată pe Serviciu (SONA − Service Oriented Network Architecture) pentru rețele de senzori ce comunică M2M. Sunt tratate aspecte privind scalabilitatea, latenţa, și irosirea resurselor sistemelor de comunicații, datorate naturii distribuite a sistemului. Soluția propusă este înglobată într-o infrastructură orientată pe serviciu, numită AAN (Application Assist Network). Structura acesteia este stratificată pe patru nivele logice, iar pentru a atinge eficieţa maximă, în stratul de rețea este folosit un controller ce are ca funcționalitate optimizarea algoritmilor și găsirea punctului cel mai apropiat de sursa comunicației pentru a fi executați. De asemenea, controllerul este în permanentă legătură cu nodurile de comunicație din rețea pentru a determina integrarea. O modalitate interesantă de reutilizare a algoritmilor este adusă în discuție prin conceptul de date "cache" în nodurile aflate „în apropierea utilizatorilor".

Pentru soluția de față, instanțierea dinamică a senzorilor sau grupului de senzori virtuali (priviți ca resurse) poate fi coordonată astfel încât comunicația să nu încarce inutil rețeaua de date. Această abordare vine totuși cu o complexitate crescută, fiind dificil de realizat în acest moment, în special în cazul în care sistemul este distribuit pe platforme aflate sub controlul a diferite entități.
Un model de soluție ce merită menționat, bazat pe arhitectura M2M a organismului de standardizare în telecomunicație ETSI (European Telecommunications Standards Institute), este prezentat în [25]. Această arhitectură este construită atât pe baza standardelor mature, verificate ale ETSI, dar și ale altor entităţi, cum ar fi IETF, 3GPP, OMA (Open Mobile Alliance) și Broadband Forum. Este făcută o analiză a unui model comercial generator de platforme M2M în cloud, fiind pus accentul pe faptul că implementările comerciale adoptate de operatorii actuali trebuie inter-relaționate și aduse la un standard comun, denumit "oneM2M". Dezvoltarea soluției în cloud cu sisteme și software standardizat aduce implicit economii de scară, dar și specificații clare ale componentelor arhitecturale, interfețelor, aplicațiilor, tehnologiilor de acces, QoS, securității și caracteristicilor de gestionare a ciclului de viață (lifecycle management).

## 5.3. Interfaţarea cu arhitecturi unificate OPC

OPC UA (OLE for process control Unified Architecture) este un protocol industrial de comunicație M2M, dezvoltat de OPC Foundation și folosit pentru interoperabilitate. Acesta este succesorul lui OPC, dar diferă semnificativ față de predecesori, deși este dezvoltat de aceeași organizație. Scopul fundației în legătură cu acest proiect a fost să furnizeze o continuare a modelului original de comunicație, OPC (bazat pe tehnologia Microsoft COM/DCOM), la o arhitectură SOA (Service Oriented Architecture) pentru controlul proceselor, independentă de platformă și, în același timp, ajutând la sporirea securității și furnizând un model informațional.

OPC UA suportă două protocoale: unul de tip binar ce implică un minim de resurse, permițând activarea simplă prin firewall-uri și un protocol de tip Web Service (SOAP) ce folosește porturile standard HTTP/HTTPS. Grație beneficiilor acestui nou protocol, se poate observa o tendinţă crescătoare a aplicațiilor industriale ce au adoptat OPC UA, atât în industriile de automatizări tradiționale OPC-centrice, cât și în cele emergente, cum ar fi cele energetice.

Modelul anterior „Classic OPC" necesită ca sistem de operare Microsoft Windows pentru a implementa funcționalitatea de server COM/DCOM. Utilizând SOA și Servicii Web, OPC UA este un sistem independent de platformă, iar prin SOAP/XML peste HTTP, OPC UA poate fi implementat într-o varietate de sisteme integrate, ce utilizează sisteme de operare în timp real deterministe (Fig. 27).

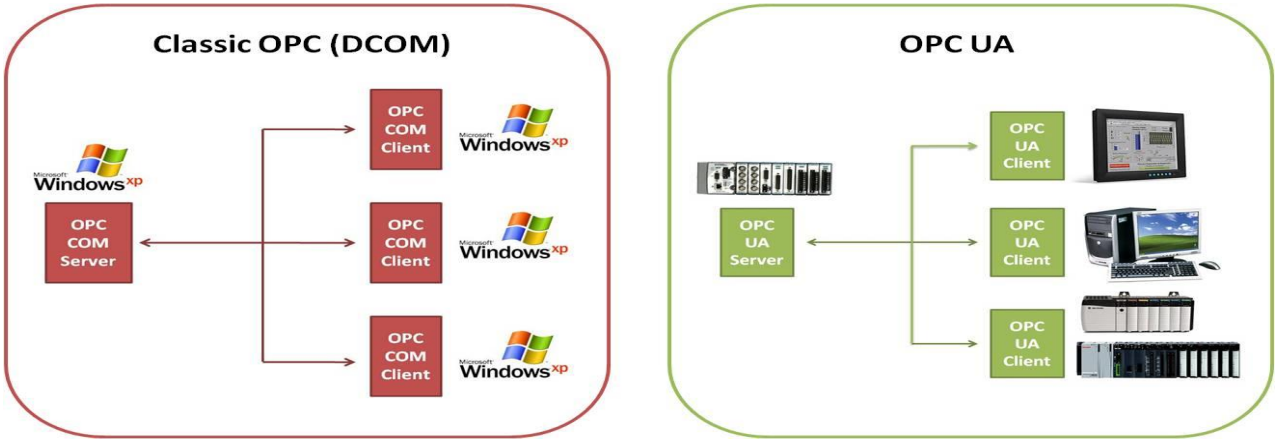



Fig. 27. OPC UA poate comunica direct cu sistemele integrate OPC UA de pe PLC

In [25] este făcută o prezentare detaliată a metamodelului OPC UA, conceptul de bază fiind nodul. Sunt definite diverse clase de noduri extinzând și specializând clasa de bază. Fiecare nod deține o mulțime fixă de atribute, depinzând de clasă, unele dintre ele fiind opționale. Relațiile dintre noduri sunt realizate prin referințe de diverse tipuri cu ierarhii extensibile. Sunt expuse, de asemenea, 34 de servicii OPC UA, grupate în seturi de servicii.

Un aspect important de considerat este procedeul de migrare a serverelor bazate pe Classic OPC (COM/DCOM) la noua tehnologie UA bazată pe Servicii Web. În mod tipic, serverele OPC bazate pe tehnologia COM utilizează interfețe proprietare pentru a accesa datele dispozitivelor în rețeaua de control. Se disting două abordări ce pot fi adoptate: împachetarea serverelor existente (wrapping) sau accesare directă a datelor dispozitivului. Împachetarea poate avea un timp mai scurt de implementare, putându-se utiliza serverele OPC existente.

DPWS (Devices Profile for Web Services) definește un set minimal de constrângeri de implementare pentru a permite dispozitivelor cu constrângeri de resurse accesul la Servicii Web sigure. În cadrul proiectului SIRENA, sub auspiciile inițiativei de cercetare Europene ITEA, Schneider Electric a produs o implementare inițială a DPWS având ca țintă dispozitivele încorporate (embedded). Aceasta a devenit ulterior open-source prin platforma SOA4D.org (SOA for Devices), de unde poate fi descărcată gratuit. De asemenea, pornind de la proiectul SIRENA,

WS4D.org (Web Services for Devices) furnizează informații și știri despre trei implementări DPWS. SODA (Service Oriented Device and Delivery Architecture) a continuat dezvoltarea și implementarea unei stive DPWS încorporate pentru dispozitive și instrumentele asociate. Actualmente, proiecte de cercetare precum SOCRADES, alcătuit din companii ca ABB, SAP, Schneider Electric și Siemens, și AESOP, se concentrează în implementarea, testarea și crearea de prototipuri de dispozitive de tip DPWS în domeniul automatizării industriale.

Tehnologiile OPC UA și DPWS au câteva puncte distinctive, cum ar fi faptul că provin din domenii disjuncte − dacă OPC UA provine din mediul industrial, DPWS s-a născut în lumea IT și a fost gândit ca o modalitate de a implementa Servicii Web pe dispozitive mici, sau că ele au fost construite având abordări distincte, ca "discovery" și "eventing", însă au un larg set de asemănări, de exemplu, amândouă sunt bazate pe SOAP. Datorită acestui fapt, este posibil să se găsească cazuri de utilizare unde cele două tehnologii pot lucra în comun sau pot beneficia una de pe urma celeilalte, iar o fuziune a acestora ar putea fi implementată într-un dispozitiv încorporat.

O cheie a succesului IoT pentru aplicațiile de automatizare industrială o reprezintă scalarea OPC UA pe dispozitive cu resurse limitate, cum sunt senzorii. Se investighează aspectul implementărilor curente ale OPC UA, prin faptul că sunt înzestrate cu caracteristici numeroase, ceea ce le face voluminoase, unele consumând chiar și 10MB de memorie. Aceasta se datorează faptului că memoria consumată este direct proporțională cu complexitatea și densitatea modelului informațional specific aplicației.

Aspectele de interfațare pentru o rețea tipică de senzori virtuali wireless (WSN) a fost descrisă și analizată în secţiunea 4.4. Este resimțită nevoia de putere înaltă de calcul, scalabilă și de infrastructuri de stocare masivă de date pentru prelucrarea, păstrarea și analiza datelor WSN, atât online cât și offline. În acest context, platforma Cloud Computing devine alegerea naturală, care furnizează o stivă flexibilă de calcul, stocare și servicii software într-o manieră scalabilă și virtualizată.

# 6. CONCLUZII

Prezentarea făcută în acest raport dovedește că obiectivele propuse pentru această etapă a proiectului, Etapa I, au fost atinse în întregime. Toate cele patru activități prevăzute au fost efectuate, iar aspectele teoretice și tehnice au fost prezentate în secțiuni distincte ale raportului. Investigațiile întreprinse se vor dovedi foarte utile pentru realizarea etapelor următoare.

Orientarea spre utilizarea standardului IEC 61499 este motivată de avantajele pe care acest standard le oferă în privința portabilității, reutilizabilității și a gradului de interes și acceptare în practica inginerească. Există zeci de publicații care se referă la acest standard, un mare număr dintre ele fiind incluse în capitolul 7. De asemenea, folosirea cloud pentru realizarea platformei colaborative și rezolvarea problemelor de modelare și optimizare, analiza riscului și prevenirea avariilor etc., cât și a soluțiilor de interfaătare prezentate, oferă șanse mari de îndeplinire a obiectivelor proiectului. Implementarea acestuia implică însă un efort considerabil, lucru vădit și de duratele mari de dezvoltare a unor proiecte similare, dar mai puțin ambițioase, cum ar fi ISaGRAF [45], sau a unor medii de calcul mai generale, cum ar fi MATLAB și Simulink [49].

Etapa I nu a presupus diseminarea explicită a unor rezultate. Totuși, în ultimele trei luni au fost publicate trei lucrări elaborate de unii membri ai echipei proiectului, cu subiecte pe tematica proiectului. Una dintre lucrări, [103], a apărut într-o revistă indexată ISI, iar alte două lucrări, [90,104], au fost publicate în „proceedings", anume la o conferință internațională care a avut loc la Sinaia și care a fost co-sponsorizată de IEEE (Institute for Electrical and Electronics Engineers). Lucrările conferiței vor fi accesibile de pe platforma IEEExplore, iar volumul de lucrări va fi indexat de ISI.

# 7. BIBLIOGRAFIE


[1] Ocheană, L.A., Rohat, O.I., Popescu, D.A. and Florea, Gh.G. "Library of Reusable Algorithms for Internet-Based Diagnose and Control System". Information Control Problems in Manufacturing, 14th IFAC Symposium on Information Control Problems in Manufacturing, 14, Part# 1, 1407-1412, 2012, http://www.ifac- papersonline.net/ Detailed/53961.html.

[2] Yang, C.-H. and Vyatkin, V. "Transformation of Simulink models to IEC 61499 Function Blocks for verification of distributed control systems". Control Engineering Practice, 20 (12), pp. 1259-1269, 2012.

[3] Aggelogiannaki, E. and Sarimveis, H. "A simulated annealing algorithm for prioritized multiobjective optimization – Implementation in an adaptive model predictive control configuration". Systems, Man, and Cybernetics, 37 (4), pp. 902-915, 2007.

[4] Rohat, O., Sistem de algoritmi reutilizabili, cu acces online, pentru controlul distribuit al proceselor industriale. Teză de doctorat, Universitatea Politehnica din Bucureşti, Facultatea Automatică şi Calculatoare, 2014.

[5] Pettey, C. „Gartner Identifies the Top 10 Strategic Technologies for 2012", www.bussinesswire.com, oct. 2011.

[6] Sünder, C., Zoitl, A., Favre-Bulle, B., Strasser, T., Steininger, H., Thomas, S. "Towards Reconfiguration Applications as basis for Control System Evolution in Zero-downtime Automation Systems", Proc. of the IPROMS NoE Virtual International Conference on Intelligent Production Machines and Systems, pp. 3-14, 2006.

[7] Rooker, M.N., Sünder, C., Strasser, T., Zoitl, A., Hummer, O., Ebenhofer, G. "Zero Downtime Reconfiguration of Distributed Automation Systems: The εCEDAC Approach", Holonic and Multi-Agent Systems for Manufacturing, Lecture Notes in Computer Science Volume 4659, 2007, pp. 326-337.

[8] Sardesai, A.R., Mazharullah, O., Vyatkin, V. „Reconfiguration of Mechatronic Systems Enabled by IEC 61499 Function Blocks", Proc. of the Australasian Conference on Robotics and Industrial Automation (ACRA), 2006.

[9] Profibus. Website: www.profibus.com.

[10] Modbus. Website: www.modbus.org.

[11] EthernetIP. Website: www.ethernet-ip.org.

[12] CAN. Website: www.can-cia.org.

[13] ControlNet. Website:www.controlnet.org.

[14] Diedrich, C., Russo, F., Winkel, L., Blevins, T. „Function Block Applications in Control Systems Based on IEC 61804", ISA Transactions, Vol. 43, Issue 1, pp. 123-31, 2004.

[15] Florea, G., Dobrescu, R., Popescu, D., Ocheana, L., Rohat, O. „PH Center a step to the next generation of Process Control Architecture", Proceedings of the 11th WSEAS International Conference on Systems Theory and Scientific Computation, aug 2011.

[16] Ferreira, P., Reyes, V., Maestre, J. „A Web-Based Integration Procedure for the Development of Reconfigurable Robotic Work-Cells", International Journal of Advanced Robotic Systems, Jan. 2013.

[17] Florea, G., Ocheană, L., Popescu, D., Rohat, O. "Emerging Technologies – The Base For The Next Goal of Process Control – Risk and Hazard Control", 11th WSEAS International Conference on Systems Theory and Scientific Computation (ISTASC '11), 2011.

[18] Colombo, A.W., Karnouskos, S., Bangemann, T. „Towards the Next Generation of Industrial Cyber-Physical Systems", Industrial Cloud-Based Cyber-Physical Systems, Springer International Publishing, 2014.

[19] Gershon, S. „Cloud Control", Rockwell Automation Publications.

[20] Lydon, B. „Internet of Things - Industrial automation industry exploring and implementing IoT", InTech Magayine, Mar-Apr 2014.

[21] Greenfield, D. „Manufacturing and the Internet of Things", AutomationWorld Magazine, Oct. 2011.

[22] Lydon, B. „Automation and Control Trends in 2013", www.automation.com, ian. 2013.

[23] Hatch, D. and Stauffer, T. "Operators on alert. Alarm standards, protection layers, HMI keys to keep plants safe". InTech, 9, 2009.
[24] Holmström, K. "Practical Optimization with the TOMLAB Environment in Matlab". Applied Optimization and Modeling Group, Mälardalen University, Västerås, Sweden, 15.09.2001, 19 pag.
[25] Lofberg, J. "YALMIP : a toolbox for modeling and optimization in MATLAB". 2004 IEEE International Symposium on Computer Aided Control Systems Design, Taipei, 4-4 Sept. 2004, pp. 284-289.
[26] Lewis, R. „Programming Industrial control systems using IEC 61131-3", The Institution of Electrical Engineers, 1995.
[27] Lewis, R. „Modelling control systems using IEC 61499: applying function blocks to distributed systems". IET, U.K. – Control Engineering Series 59, 2001.
[28] Vyatkin, V. "IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review". IEEE Transactions on Industrial Informatics, 7 (4), pp. 768-781, 2011.
[29] Thramboulidis, K., Sierla, S., Papakonstantinoul, N., Koskinen, K. "An IEC 61499 based approach for distributed batch process control," 5th IEEE International Conference on Industrial Informatics, 2007, pp. 177-182, 2007.
[30] Dimitrova, D., Panjaitan, S., Batchkova, I., Frey, G. "IEC 61499 Component Based Approach for Batch Control Systems", Proceedings of the 17th World Congress IFAC, Seoul, Korea, July 6-11, 2008.
[31] Lepuschitz, W., Zoitl, A. "Integration of a DCS based on IEC 61499 with Industrial Batch Management Systems," 13th IFAC Symposium on Information Control Problems in Manufacturing, Moscow, 2009.
[32] Lastra, J.L.M., Lobov, A., Godinho, L. "Closed loop control using an IEC 61499 application generator for scan-based controllers", 10th IEEE Conference on Emerging Technologies and Factorz Automation (ETFA), 2005.
[33] Aendenroomer, A., He, H., Ling, K.V., Sreenidhi, K.A., Mullamitha, M.A., Goh, K.M. "Closed-loop Modeling and Rapid ApplicationGeneration, using IEC 61499 Function Blocks and XML", 3rd Int. Conf. on Reconfigurable Manufacturing, 10-12 May 2005.
[34] Christensen, J.H., Strasser, T., Valentini, A., Vyatkin, V., Zoitl, A. "The IEC 61499 Function Block Standard: Software Tools and Runtime Platforms", ISA Automation Week 2012.
[35] Pollard, J. "A look at IEC 61499", Control Design Magazine, oct. 2007.
[36] Bezák, T. "Usage of IEC 61131 and IEC 61499 Standards for Creating Distributed Control Systems", Scientific Monographs in Automation andcomputer Science, Vol. 3, 2012.
[37] Kepware Technologies. Website: www.kepware.com.
[38] Matrikon OPC Servers. Website: www.matrikonopc.com.
[39] Christensen, J.H. „IEC 61499 Architecture, Engineering Methodologies and Software Tools", Knowledge and Technology Integration in Production and Services, Volume 101, 2002, pp 221-228.
[40] Heverhagen, T., Hirschfeld, R., Trach, R. „Introduction to function blocks", Website: www.functionblocks.org.
[41] Strasser, T., Christensen, J.H., Valente, A., Chouinard, J., Carpanzano, E., Valentini, A., Mayer, H., Vyatkin, V., Zoitl, A. "The IEC 61499 Function Block Standard: Launch and Takeoff", ISA Automation Week 2012.
[42] Hegny, I., Zoitl, A., Lepuschitz, *W.* "Integration of simulation in the development process of distributed IEC 61499 control applications", Proceedings of IEEE International Conference on Industrial Technology, 2009.
[43] DEIF. Wind Power Technology. From Control Design to PLC Within Minutes. Website: www.deifwindpower.com
[44] CODESYS. Website: www.codesys.com
[45] ISAGRAF. Website: www.isagraf.com
[46] Ebenhofer, G., Rooker, M., Falsig, S. „Generic and Reconfigurable IEC 61499 function blocks for advanced platform independent engineering", Proceeding of 9th IEEE International Conference on Industrial Informatics (INDIN), 2011.
[47] Hegny, I., Strasser, T., Melik-Merkumians, M., Wenger, M., Zoitl, A. „Towards an Increased Reusability of Distributed Control Applications Modeled in IEC 61499", IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA), 2012.
[48] Holobloc Inc., Resources for the New Generation of Automation and Control, Function Block Development Kit (FBDK). Website: www.holobloc.com.
[49] MATLAB — The language of technical computing. Website:www.mathworks.com/products/ matlab/
[50] LabVIEW System Design Software. Website: www.ni.com/labview/
[51] Benner, P., Mehrmann, V., Sima, V., Van Huffel, S. and Varga, A. "SLICOT – A Subroutine Library in Systems and Control Theory". In: B.N. Datta (Ed.), Applied and Computational Control, Signals, and Circuits, 1, ch. 10, pp. 499-539, 1999.
[52] Dongarra, J.J., Du Croz, J., Duff, I.S., Hammarling, S. „Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms", ACM Trans. Math. Softw., 1990.

[53] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov , S., Sorensen, D. „LAPACK Users' Guide: Third Edition", Society for Industrial and Applied Mathematics, 1999.
[54] ThinkCycle. Website: p2pfoundation.net/ThinkCycle.
[55] NxtControl. Website: http://www.nxtcontrol.com/en.html
[56] Strasser, T., Auinger , F., Zoitl, A. „Development, implementation and use of an IEC 61499 function block library for embedded closed loop control", 2nd IEEE International Conference on Industrial Informatics INDIN 2004, pp. 594 – 599.
[57] Wei, Y. „Implementation of IEC 61499 Distributed Function Block Architecture for Industrial Measurement and Control Systems (IPMCS)", Total Enterprise Solutions Conference, ICAM, 2001.
[58] FBDK, "Resources for the New Generation of Automation and Control. Function Block Development Kit (FBDK)". Website: www.holobloc.com.
[59] 4DIAC, Framework for Distributed Industrial Automation and Control. Website: www.fourdiac.com.
[60] Sima, V. "Subspace-Based Algorithms for Multivariable System Identification". Studies in Informatics and Control, 5 (4), 335-344, 1996.
[61] Sima, V. and Van Huffel, S. "Efficient Numerical Algorithms and Software for Subspace-Based System Identification". In: Proceedings of the 2000 IEEE International Conference on Control Applications and IEEE International Symposium on Computer-Aided Control Systems Design, September 25–27, 2000, Anchorage, Alaska, U.S.A., pp. 1–6. Omnipress, 2000.
[62] Mastronardi, N., Kressner, D., Sima, V., Van Dooren, P. and Van Huffel, S. "A Fast Algorithm for Subspace State-Space System Identification via Exploitation of the Displacement Structure". J. Comput. Appl. Math., 132 (1):71-81, 2001.
[63] Sima, V., Sima, D.M. and Van Huffel, S. "SLICOT System Identification Software and Applications". In: Proceedings of the 2002 IEEE International Conference on Control Applications and IEEE International Symposium on Computer Aided Control System Design, CCA/CACSD 2002, September 18-20, 2002, Glasgow, Scotland, U.K., pp. 45-50. Omnipress, 2002.
[64] Sima, V. "Fast Numerical Algorithms for Wiener Systems Identification". In V. Barbu, I. Lasiecka, D. Tiba, C. Varsan (Eds.), Analysis and Optimization of Differential Systems, Kluwer Academic Publishers, Boston/Dordrecht/ London, pp. 375–386, 2003.
[65] Sima, V. "Efficient Data Processing for Subspace-Based Multivariable System Identification". In: Proceedings of IFAC Workshop on Adaptation and Learning in Control and Signal Processing (ALCOSP 2004), and IFAC Workshop on Periodic Control Systems (PSYCO 2004), August 30th - September 1st, 2004, Yokohama, Japan, pp. 871-876, 2004.
[66] Sima, V., Sima, D.M. and Van Huffel, S. "High-Performance Numerical Algorithms and Software for Subspace-Based Linear Multivariable System Identification". J. Comput. Appl. Math., 170 (2), 371-397, 2004.
[67] Sima, V. Algorithms for Linear-Quadratic Optimization. Pure and Applied Mathematics: A Series of Monographs and Textbooks, vol. 200, Marcel Dekker, Inc., New York, 1996.
[68] Sima, V. "Computational Experience in Solving Algebraic Riccati Equations". In: Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC'05, 12-15 December 2005, Seville, Spain, pp. 7982-7987. Omnipress, 2005.
[69] Van Huffel, S., Sima, V., Varga, A., Hammarling, S. and Delebecque, F. "High-Performance Numerical Software for Control". IEEE Control Syst. Mag., 24 (1), 60-76, 2004.
[70] Benner, P., Kressner, D., Sima, V. and Varga, A. „Die SLICOT-Toolboxen für Matlab". at–Automatisierungstechnik, 58 (1):15–25, 2010.
[71] Sima, V. "Recent Developments in SLICOT Library and Related Tools". In: Proceedings of the 15th International Conference on System Theory, Control and Computing, pp. 564-569, 2011.
[72] Sima, V., Tits, A. and Yang, Y. "Computational Experience with Robust Pole Assignment Algorithms". In: Proceedings of the 2006 IEEE International Conference on Control Applications (CCA), 2006 IEEE Conference on Computer-Aided Control Systems Design (CACSD), and 2006 IEEE International Symposium on Intelligent Control (ISIC), Technische Universität München, Munich, Germany, October 4-6, 2006, pp. 36-41. Omnipress, 2006.
[73] Benner, P., Sima, V., Voigt, M. $L_\infty$-norm computation for continuous-time descriptor systems using structured matrix pencils. IEEE Trans. Automat. Contr., 57(1), pp.233-238, 2012.
[74] Isermann, R. "Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes". Control Engineering Practice, 5 (5), pp. 709-719, 1997.

[75] Jiang, J. "Fault-Tolerant Control Systems – An Introductory Overview". Acta Automatica Sinica, 31 (1), pp. 161-174, 2005.
[76] Zhang, Y. and Jiang, J. "Bibliographical Review on Reconfigurable Fault-Tolerant Control Systems". Annual Reviews in Control, 32 (2), pp. 229-252, 2008.
[77] Hwang, I., Kim, S., Kim, Y. and Seah, C.E. "A Survey of Fault Detection, Isolation and Reconfiguration Methods". IEEE Transactions on Control Systems Technology, 18 (3), pp. 636-653, 2010.
[78] Zolghadri, A. "The Challenge of Advanced Model-Based FDIR Techniques for Aerospace Systems: The 2011 Situation". In: 4th European Conference for Aerospace Sciences, 2011.
[79] Venkatasubramanian, V., Rengaswamy, R., Yin, K. and Kavuri, S.N. "A Review of Process Fault Detection and Diagnosis. Part I. Quantitative Model-Based Methods". Computers and Chemical Engineering, 27 (3), pp. 293–311, 2003.
[80] Venkatasubramanian, V., Rengaswamy, R. and Kavuri, S.N. "A Review of Process Fault Detection and Diagnosis. Part II. Qualitative Models and Search Strategies". Computers and Chemical Engineering, 27 (3), pp. 313–326, 2003.
[81] Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N. and Yin, K. "A Review of Process Fault Detection and Diagnosis. Part III. Process History Based Methods". Computers and Chemical Engineering, 27 (3), pp. 327–346, 2003.
[82] Magni, J.F., Bennani, S. and Terlouw, J. (Eds.). Robust Flight Control: A Design Challenge. Edited by Springer-Verlag, 1997.
[83] Blanke, M., Kinnaert, M., Lunze, J. and Staroswiecki, M. Diagnosis and Fault Tolerant Control, 2nd ed. Springer, 2006.
[84] Lunze, J. and Richter, J.H. "Reconfigurable Fault-Tolerant Control: A Tutorial Introduction". European Journal of Control, 14 (5), pp. 359–386, 2008.
[85] Verhaegen, M., Kanev, S., Hallouzi, R., Jones, C., Maciejowski, J. and Smaili, H. "Fault Tolerant Flight Control – A Survey". Chapter 2 in: Fault-Tolerant Flight Control – A Benchmark Challenge, C. Edwards, T. Lombaerts, H. Smaili (Eds.), Springer-Verlag, pp. 47-89, 2010.
[86] Ciubotaru, B.D., Staroswiecki, M. and Christov, N.D. „Modified Pseudo-Inverse Method with Generalized Linear Quadratic Regulator for Fault Tolerant Model Matching with Prescribed Stability Degree". In: Proceedings of the 50th IEEE Conference on Decision and Control, and the 11th European Control Conference, paper MoC03.5, 2011.
[87] Ciubotaru, B.D. and Staroswiecki, M. "Extension of Modified Pseudo-Inverse Method with Generalized Linear Quadratic Stabilization". In: Proceedings of the 29th American Control Conference, pp. 6222-6224, 2010.
[88] Ciubotaru, B.D. and Staroswiecki, M. „Comparative Study of Matrix Riccati Equation Solvers for Parametric Faults Accommodation". In: Proceedings of the 10th European Control Conference, pp. 1371-1376, 2009.
[89] Sima, V., Benner, P. (2014). "Numerical Investigation of Newton's Method for Solving Continuous-time Algebraic Riccati Equations". Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2014), Vienna, Austria, 1-3 September, 2014 (CD-ROM), Vol. 1, pp. 404-409.
[90] Sima, V. "Efficient Computations for Solving Algebraic Riccati Equations by Newton's Method". In: Proceedings of the 2014 18th International Conference on System Theory, Control and Computing, October 17-19, 2014, Sinaia, Romania, pp.609-614, 2014.
[91] Tajer, A., Veeravalli, V.V., Poor H.V. Outlying Sequences Detention in Large Data Sets. IEEE Signal Processing Magazine, Vol. 31, No. 5, Sep. 2014, pp. 44-56.
[92] Schubert, L., Jeffery, K. (Eds.). "Advances in Clouds – Research in Future Cloud Computing". Expert Group Report Public version 1.0, European Union, 2012, http://cordis.europa.eu/fp7/ict/ssai/docs/future-cc-2may-finalreport-experts.pdf
[93] Kriz, P. "Survey on open source platform-as-a-service solutions for education." Proceedings of the 18th International Database Engineering & Applications Symposium. ACM, 2014.
[94] Zhang, Z., Wu, C., and Cheung, D.W.L. "A survey on cloud interoperability: taxonomies, standards, and practice." ACM SIGMETRICS Performance Evaluation Review 40.4 (2013): 13-22.
[95] Cohen, R. "Examining cloud compatibility, portability and interoperability."ElasticVapor: Life in the Cloud, DOI: http://www. elasticvapor. com/2009/02/examining-cloudcompatibility. Html (2009).
[96] Ranabahu, A., and Sheth, A. "Semantics centric solutions for application and data portability in cloud computing." Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010.

[97] Swan, C. Docker drops LXC as default execution environment. InfoQ http://www.infoq.com/news/2014/03/docker_0_9 (2014)

[98] Chamberlain, R., Invenshure, L.L.C., and Schommer, J. Using Docker to support reproducible research. Technical Report 1101910, figshare, 2014. http://dx. doi. org/10.6084/m9. Figshare. 1101910, 2014.

[99] Von Eicken, T. Docker vs. VMs? Combining Both for Cloud Portability Nirvana. RightScale Blog. http://www.rightscale.com/blog/cloud-management-best-practices/docker-vs-vms-combining-both-cloud-portability-nirvana (2014)

[100] Boettiger, C. An introduction to Docker for reproducible research, with examples from the R environment. arXiv preprint arXiv:1410.0846 (2014).

[101] Florea, G.., Dobrescu, R. "Risk and Hazard Control the new process control paradigm". Systems, Control, Signal Processing and Informatics II, Prague, 2014.

[102] Florea, G. "Contribuţii la proiectarea unor arhitecturi de conducere evoluată a proceselor industriale." Teză de doctorat, Universitatea Politehnica din Bucureşti, Facultatea Automatică şi Calculatoare, 2014.

[103] Rohat, O., Popescu, D. „Web System for the Remote Control and Execution of an IEC 61499 Application", Studies in Informatics and Control (SIC), Issue 3, Vol. 23, 2014.

[104] Rohat, O., Popescu, D. "Is FBDK suitable for develping and implementing process control optimization problems?", 18th International Conference on System Theory, Control and Computing, Sinaia, October 17-19, 2014.