

REZUMAT:

Documentație de proiectare arhitectură software pentru sisteme de realitate augmentată in-vehicle

Proiect complex:

PN-III-P1-1.2-PCCDI-2017-0917

Proiect component:

**P2 - Comunicații eficiente bazate pe dispozitive inteligente în scenarii
interactive de realitate augmentată pentru autovehicule**

Parteneri:

Universitatea Ovidius din Constanța

Universitatea Ștefan cel Mare din Suceava

Autori:

Universitatea Ovidius din Constanța

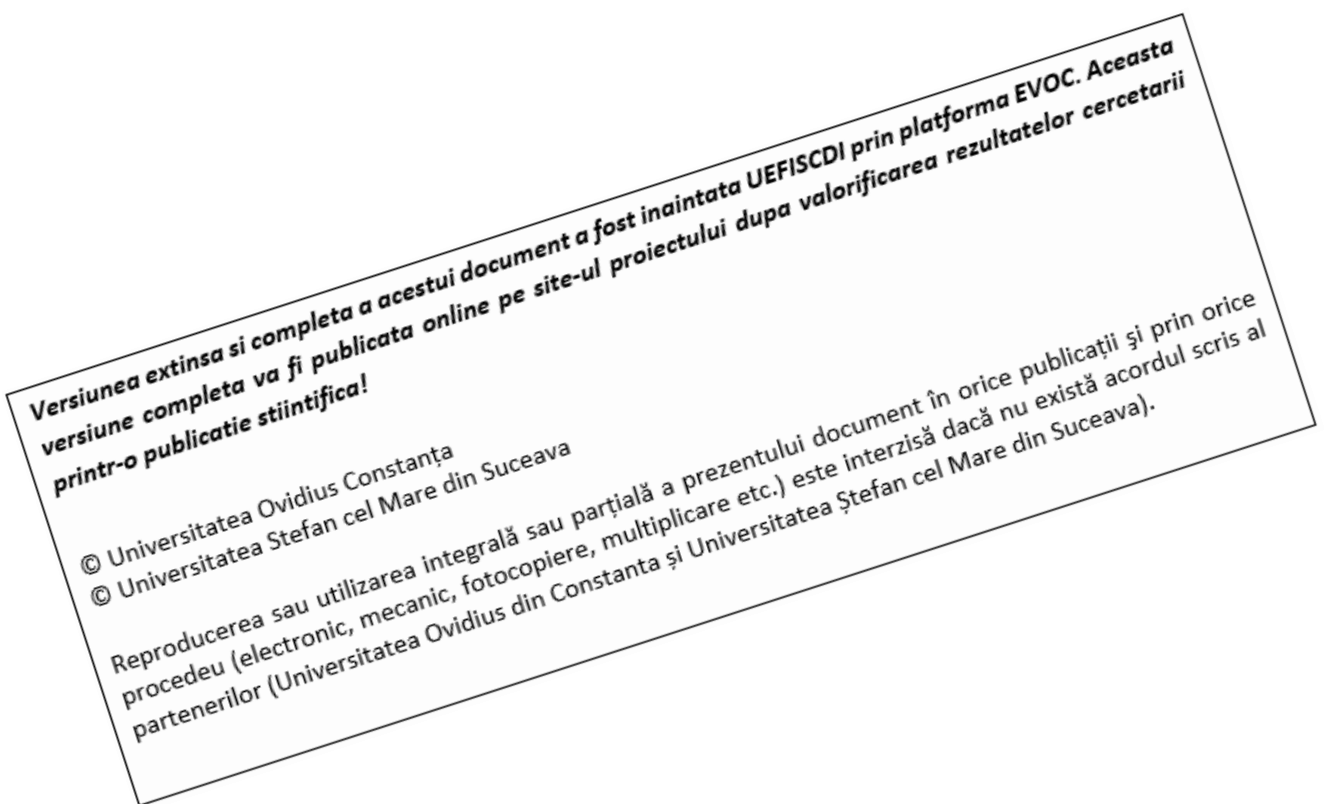
Prof.univ.dr. Dorin Mircea POPOVICI
Conf.univ.dr. Dragoș-Florin SBURLAN
Conf.univ.dr. Crenguța Mădălina PUCHIANU
Lect.univ.dr. Elena BĂUTU

Universitatea Ștefan cel Mare din Suceava

Prof.dr.ing. Radu-Daniel VATAVU
Prof.dr.ing. Ștefan-Gheorghe PENTIUC
Conf. univ. dr.ing. Ovidiu-Andrei SCHIPOR

Cuprins

Cuprins.....	2
1. Introducere.....	3
2. Proiectarea si implementarea arhitecturii Euphoria.....	3
2.1. Cerințe de proiectare și criterii de performanță	4
2.2. Proiectarea arhitecturii software Euphoria.....	5
3. Proiectarea si implementarea de componente software care comunica cu server-ul Euphoria.....	9
4. Referințe.....	11



1. Introducere

Etapa aferentă anului 2018 de implementare a proiectului component P2 a constatat în dezvoltarea unei infrastructuri software pentru sisteme de realitate augmentată in-vehicle, în conformitate cu Activitatea 1.2 „*Dezvoltare infrastructura software pentru sisteme de realitate augmentată in-vehicle*” din planul de realizare a proiectelor componente, activitate de tip A2 (cercetare industrială). Scopul dezvoltării infrastructurii software este de a oferi suportul necesar proiectării, implementării și evaluării modulelor și componentelor planificate pentru următoarele etape ale proiectului component P2.

Activitățile tehnice și științifice întreprinse în acest sens în cadrul etapei anului 2018 au constatat în:

- Proiectarea unei arhitecturi software bazată pe procesare de evenimente denumită Euphoria (Event-based Unified Platform for HeteRogeneous and Aynchronous Interactions) cu un grad ridicat de generalitate necesar pentru a putea integra diverse tipuri de dispozitive de intrare și ieșire. În conformitate, am pregătit documentația tehnică de proiectare a arhitecturii software.
- Implementarea arhitecturii software Euphoria folosind tehnologii web consacrate și validate în mediul industrial.
- Proiectarea și implementarea de componente software care trimit/primesc mesaje către/de la server-ul Euphoria.
- Evaluarea performanței tehnice a arhitecturii software Euphoria în ceea ce privește viteza de procesare a evenimentelor, respectiv gradul de scalabilitate. În conformitate, am pregătit documentația de testare a arhitecturii.
- Pregătirea, redactarea, trimiterea în vederea evaluării și publicării de lucrări științifice în tematica proiectului component P2.

Prezentul document reprezintă rezumatul raportului de proiectare din cadrul etapei 1 – 2018.

2. Proiectarea și implementarea arhitecturii Euphoria

Acest raport prezintă în rezumat detaliile de proiectare și implementare pentru **Euphoria (Event-based Unified Platform for Heterogeneous and Asynchronous Interactions)**, o nouă arhitectură software pentru realizarea de interacțiuni flexibile, eterogene și asincrone cu conținut digital în noi medii inteligente și contexte de aplicare privind realitatea augmentată,

cum ar fi sistemele de realitate augmentată in-vehicle adresate în cadrul proiectului component P2. Euphoria implementează producția, transmiterea, procesarea și consumul de evenimente folosind protocoale și tehnologii web consacrate de către mediul industrial (cum ar fi HTTP, WebSockets, node.js, JSON, etc.) și care sunt suportate nativ de o varietate mare de dispozitive conectate, cum ar fi PC-uri, tablete și telefoane smart, ceasuri smart, etc.

Euphoria este disponibilă online la adresa web permanentă

<http://www.eed.usv.ro/mintviz/resources/Euphoria>

2.1. Cerințe de proiectare și criterii de performanță

În proiectarea arhitecturii Euphoria am considerat o serie de criterii și cerințe de performanță, descrise în detaliu în această secțiune. De asemenea, în scopul unei poziționări clare și exacte a arhitecturii software Euphoria în contextul literaturii de specialitate prezentăm în continuare o analiză comparativă cu privire la arhitecturile software existente.

Arhitectura software Euphoria a fost proiectată pentru a integra ușor dispozitive și aplicații software într-un mediu inteligent, alcătuit dintr-o diversitate de consumatori și producători, cu scopul specific de a putea fi aplicată pentru scenarii de utilizare privind interacțiuni utilizator în vehicule smart (connected cars). Prin urmare, am adoptat mai multe tehnici de procesare și proprietăți de calitate ca și cerințe pentru arhitectura Euphoria; vezi figura 1 pentru o privire de ansamblu asupra acestora.

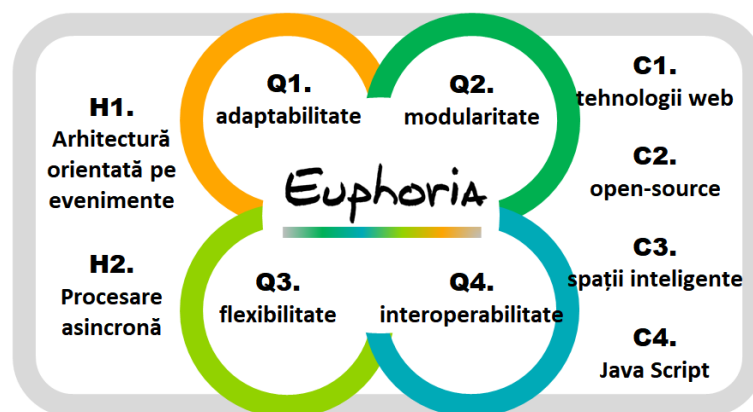


Figura 1. Reprezentarea vizuală a celor două tehnici de procesare (H1 și H2), a cele patru proprietăți de calitate (Q1, Q2, Q3 și Q4) și, respectiv, a cele patru proprietăți contextuale (C1, C2, C3 și C4) adoptate pentru proiectarea arhitecturii software Euphoria.

2.2. Proiectarea arhitecturii software Euphoria

În această secțiune prezentăm modelul arhitecturii software euphoria. Figura 2 ilustrează designul adoptat pentru Euphoria cu componentele principale: producători, emițători, motor de procesare, receptori și consumatori, în conformitate cu modelul arhitecturilor bazate pe evenimente [19]. Am proiectat Euphoria cu o separare puternică între blocurile sale funcționale (Q2) și cu o dimensiune cât mai redusă a straturilor de adaptare (Q1).

De asemenea, respectând principiile de design H1 [18,19,20], informațiile despre evenimentele declanșate de către producători sunt colectate, ambalate în mesaje și expuse de către emițători către următorul strat. Ulterior, mesajele sunt procesate și expediate de către motorul de procesare către receptori și consumatori, care vor interpreta mesajele respective în conformitate cu logica lor internă și vor lua acțiunile corespunzătoare în cadrul mediului inteligent. De exemplu, un eveniment detectat de un ceas smart purtat de un șofer al unei mașini smart la încheietura mâinii va duce la selectarea unui element pe un ecran. Un mesaj transmis prin Euphoria constă într-un antet cu metadate (de exemplu, numele dispozitivului, adresa IP și numele evenimentului) și un corp de mesaj care conține datele relevante pentru eveniment. Figura 2 ilustrează patru exemple de mesaje de diverse complexități produse de o brățară, un smartphone și doi senzori de detecție a mișcării.

Producătorul reprezintă componenta generatoare a evenimentelor (o modificare relevantă a stării este notificată arhitecturii sub forma unui eveniment). Producătorii sunt instanțiați de dispozitivele de intrare (de exemplu, senzori de mișcare, dispozitive mobile inteligente, dispozitive portabile, etc.) sau de servicii software (de exemplu, o comandă gestuală a fost detectată pentru un anumit utilizator de către un serviciu de recunoaștere a gesturilor de pe web).

Pentru producătorii instanțe ale dispozitivelor de intrare, evenimentele se referă la orice schimbare fizică a stărilor acestor dispozitive care este relevantă pentru aplicație, cum ar fi apăsarea unui buton sau o schimbare în orientarea încheieturii mâinii detectată de un ceas smart. Similar, evenimentele software pot fi produse în legătură cu schimbările în starea unei anumite aplicații.

venimentele software se dovedesc a fi utile pentru fazele de depanare, simulare sau pentru integrarea logicii complexe furnizate de bibliotecile și serviciile terțe, cum ar fi cele disponibile pe web. Caracteristic pentru arhitecturile bazate pe evenimente este faptul că straturile distincte nu cunosc detaliile de funcționare ale celorlalte straturi ale arhitecturii (Q2).

```

message{2} ▼
  header{3} ▶ deviceName      :MYO Armband #1
               deviceIP       :192.168.0.9
               eventName      :Gesture
               timestamp       :1493376106270
  body {3} ▶ gesture          :TwistRight
            acceleration{3} ▶ x      : 1.221
                           y      : 0.613
                           z      :-0.121
            orientation {3} ▶ azimuth : 0.196
                           pitch   :-0.184
                           roll    : 1.254

message{2} ▼
  header{3} ▶ deviceName      :Smart Phone #3
               deviceIP       :192.168.0.5
               eventName      :Touch-Select
               timestamp       :1493376183594
  body {4} ▶ touchTime        :267
            touchPressure     :0.62
            touchLocation {2} ▶ x:   273
                           y:   361
            deviceLocation{3} ▶ x: 2425
                           y:  578
                           z: 1587

message{2} ▼
  header{3} ▶ deviceName      :Vicon #1
               deviceIP       :192.168.0.7
               eventName      :Right-Hand
               timestamp       :1493376183681
  body {3} ▶ modelInfo{3} ▼
            name              :Right-Hand
            pointingMarker    :A
            orientationSegment:ABCD
            markers {4} ▼
            A{1} ▶ location {3}▶...
            B{1} ▶ ...
            C{1} ▶ ...
            D{1} ▶ ...
            segments {1} ▼
            ABCD{2} ▶ markers: [A,B,C,D]
                   ▶ orientation{3}▶...

message{2} ▼
  header{3} ▶ deviceName      :Kinect #1
               deviceIP       :192.168.0.6
               eventName      :Bob-Skeleton
               timestamp       :1493376182723
  body {3} ▶ joints{25} ▼
            SpineBase {1}▶location {3} ▶...
            SpineMid  {1}▶location {3} ▶...
            Neck      {1}▶location {3} ▶...
            ...
            ThumbRight{1}▶location {3} ▶...
            bones {14} ▼
            ArmUpLeft {2}▶joints      :[...]
                           orientation{3}▶...
            ...
            FootRight {2}▶joints      :[...]
                           orientation{3}▶...

```

Figura 2. Exemple de mesaje JSON de diferite complexitati: un mesaj ce contine informatii despre acceleratie si orientarea vehiculului colectate de o bratara Myo (stanga sus); informatii touch primite de la un telefon smartphone (dreapta sus); informatii despre miscare obtinute de la un sistem Vicon (stanga jos); un mesaj ce contine informatii de la un schelet Microsoft Kinect (dreapta jos).

De exemplu, nu este important ca producătorii să cunoască care componente vor procesa și consuma evenimentele pe care aceștia le produc. Această caracteristică facilitează integrarea dispozitivelor și a aplicațiilor software noi în Euphoria, inclusiv a celor care urmează să apară în viitor, cum ar fi dispozitive pentru realitatea augmentată mobilă [61], fără modificări ale principalelor componente ale arhitecturii Euphoria.

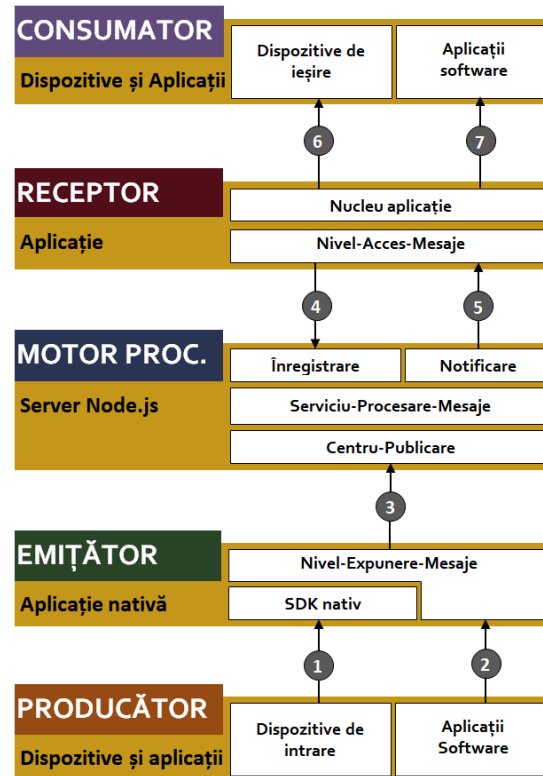


Figura 3. Arhitectura Euphoria bazată pe evenimente pentru proiectarea interacțiunilor în medii inteligente. Euphoria este formată din producători de evenimente, emițători, un motor de procesare, receptori și consumatori. Numerele de la ① la ⑦ reprezintă fluxurile de date discutate în text.

Odată ce un eveniment a fost produs, se declanșează instanțierea unui mesaj (fluxurile de date ① și ② din Figura 3), care se deplasează către nivelurile următoare ale arhitecturii. În cazul în care evenimentul a fost produs de un dispozitiv de intrare, emițătorul este construit în jurul SDK-ului dispozitivului respectiv. Acesta oferă componentele software necesare pentru a colecta parametrii specifici evenimentului (de exemplu, proprietățile unei atingeri efectuate pe un ecran tactil, orientarea unui dispozitiv de detectare a mișcării etc.).

Emițătorul colectează datele evenimentului și le împachetează sub formă de mesaje care sunt livrate către straturile superioare ale arhitecturii. Un mesaj constă dintr-un antet cu meta-date (i.e., tipul evenimentului și identificarea producătorului) și un corp de mesaj cu detalii despre schimbarea stării (e.g., butonul care a fost apăsat, datele de orientare a dispozitivului, tipul gestului efectuat de către utilizator, coordonatele de poziție de la un senzor de mișcare, etc.). În timp ce antetul mesajului are o structură fixă, corpul său depinde de tipul dispozitivului și de

evenimentul produs. În cele mai multe cazuri, stratul de expunere a mesajelor (Figura 3) trebuie implementat în cod nativ, deși unele SDK-uri sau aplicații software pot avea deja opțiuni de interfațare pentru un anumit dispozitiv de intrare.

Emițătorii și receptorii reprezintă interfața dintre nucleul Euphoria și varietatea de dispozitive și aplicații software pe care dorim să le conectăm. Pentru a expune sau a primi un mesaj, este nevoie de o conversie software între un model centrat pe producător sau consumator și un model generic, specific pentru Euphoria. Utilizarea tehnologiilor robuste și optimizate, cum ar fi HTTP sau JSON, conduce la un strat de adaptare cât mai eficient, păstrând în același timp flexibilitatea și interoperabilitatea. De fapt, emițătorii și receptorii sunt singurele niveluri ale arhitecturii care trebuie implementate de către dezvoltatori.

Deoarece componenta emițător este necesară pentru ca fiecare dispozitiv de intrare să fie integrat în aplicațiile care vor funcționa în mediul inteligent, am proiectat-o generală și reutilizabilă conform proprietăților de calitate Q3 și Q4.

În formă brută, emițătorul constă doar în codul necesar pentru încapsularea proprietăților specifice într-o reprezentare standard JSON și pentru trimiterea acestui mesaj printr-o cerere HTTP. Alimentarea arhitecturii noastre cu un eveniment este la fel de simplă precum o cerere HTTP către o adresă URL: șirul de interogare reprezintă chiar mesajul evenimentului.

Motorul de procesare este componenta esențială a arhitecturii Euphoria. Principala sa responsabilitate este de a primi mesaje de la emițători (a se vedea fluxul de date ③ din Figura 3) și de a le transmite către consumatorii corespunzători care s-au înregistrat pentru a primi acele evenimente. În implementarea noastră, motorul de procesare rulează sub forma unui server node.js. Mesajele evenimentului sunt despachetate de către serviciul de procesare a mesajelor (Figura 3).

Înainte de a primi o notificare, un consumator trebuie să se înregistreze, urmând un protocol standard. Ca parte a acestui protocol, consumatorul va specifica tipul de evenimente pe care vrea să le recepționeze și producătorii corespunzători.

Odată ce înregistrarea este finalizată, motorul de procesare va anunța automat consumatorul cu privire la acele evenimente care îndeplinesc cerințele convenite în timpul înregistrării. Pentru efectuarea atât a înregistrării, cât și a notificării, motorul de procesare și consumatorul folosesc tehnologia WebSockets, un set de protocoale de comunicații care permit sesiuni TCP full-duplex între un client și un server [62,63].

Serviciul de procesare a mesajelor (Figura 3) dirijează mesajele către consumatorii corespunzători și, de asemenea, transformă datele în formatul interschimbabil JSON (JavaScript Object Notation) (Q4). Motorul de procesare nu cunoaște structura internă a producătorilor și a consumatorilor. Această abstractizare susține atât scalabilitatea (Q1), cât și capacitatea de integrare a dispozitivelor de intrare eterogene (Q4).

Receptorul este reprezentat de orice aplicație software terță care solicită evenimente din mediul inteligent pentru a le folosi în logica sa internă. Receptorul stabilește o conexiune WebSockets cu motorul de procesare folosind Nivelul-Acces-Mesaje (vezi fluxurile de date ④ și ⑤ din Figura 3).

În plus față de lista de filtre pentru tipurile de evenimente și de producători, receptorul poate specifica și alți parametri opționali, cum ar fi un interval de timp minim între primirea a două evenimente consecutive de același tip pentru a preveni acumularea excesivă a evenimentelor.

Această practică servește de asemenea la prevenirea supraîncărcării rețelei cu mesaje inutile sau redundante. Un obiectiv important al acestei arhitecturi este acela de a menține un strat foarte subțire între receptor și motorul de procesare, astfel încât orice aplicație software existentă să se poată integra cu Euphoria cu un efort minim de programare (proprietăți de calitate Q1 și Q2). În acest scop, am decis să implementăm Nivelul-Acces-Mesaje folosind tehnologia WebSockets, permițând astfel transferul de date în timp real și asincron (vezi tehnica de H1) între motorul de procesare și receptor.

Odată primit, mesajul declanșează o acțiune specifică asupra consumatorului, care reprezintă al cincilea nivel al arhitecturii (vezi fluxurile de date ⑥ și ⑦ din figura 2). Acest nivel constă în dispozitive de ieșire și aplicații software cu care utilizatorii interacționează direct. Consumatorul reprezintă interfața de ieșire dintre Euphoria și mediul exterior. Detaliile implementării consumatorilor sunt izolate de către receptori pentru a face motorul de procesare cât mai decuplat posibil.

3. Proiectarea și implementarea de componente software care comunica cu server-ul Euphoria

Aplicațiile producător (*GenericProducer*), respectiv consumator (*GenericConsumer*) de evenimente proiectate, implementate și testate în această etapă au fost de notificare a conducătorului auto în timpul conducerii vehiculului. Folosind prototipul implementat,

conducătorul poate alege să primească mesaje ce conțin informații despre starea vremii, respectiv poate bloca notificari primite de la Facebook, Gmail, WhatsApp sau lansate automat de existența actualizărilor aplicațiilor instalate pe telefonul mobil. În acest mod, conducătorul auto este notificat numai cu informații importante pentru el, atenția lui fiind astfel cât mai puțin distrasă de la activitatea de condus.

Aplicația *GenericProducer* inglobează, în acest moment, 3 tipuri de producători de conținut:

- un producător de conținut generic, denumit *GenericComponent*, personalizabil, ce furnizează notificări/mesaje de dimensiuni customizabile, la intervale de timp stabilite dinamic. Acest producător și-a dovedit utilitatea în contextul testării aplicației din punct de vedere al gestiunii timpului și resurselor platformei.
- un producător de conținut, denumit *NotificationComponent*, ce preia de la dispozitivul mobil al conducătorului auto toate notificările, pe măsura ce acestea apar, apoi le împachetează într-o manieră unitară și le trimite sub forma unui mesaj în format JSON către serverul Euphoria.
- un producător de conținut, denumit *WeatherComponent*, ce generează mesaje ce folosesc drept sursă informații despre vreme. Acesta identifică locația geografică a dispozitivului mobil al conducătorului auto. Ulterior se conectează la un serviciu de vreme disponibil gratuit online. În final, acesta produce, în aplicație, notificări regulate, la intervale de timp personalizabile, cu informații despre vremea în locația curentă a telefonului conducătorului auto aflat în trafic.

Aplicația *GenericConsumer* primește de la Euphoria obiecte JSON și afișează pe display-ul consumatorului informații despre acestea. Momentan, scopul afișării a fost de a testa fiabilitatea unei astfel de abordări. Pentru afișarea informației produse de către producător, am avut în vedere în primul rând vizibilitatea informației în diferite condiții de mediu, precum și minimizarea perturbării atenției participanților la trafic în general, respectiv a conducătorului auto, în particular, de către afișarea acestei informații.

Fiecare aplicație a fost proiectată folosind stilul arhitectural condus de evenimente ce se bazează pe modelul de proiectare Observer, varianta broadcasting [66]. Am aplicat acest stil pentru a avea componente independente ce procesează asincron datele.

Fiecare producator/consumator de evenimente a fost încapsulat într-un pachet ce a fost proiectat obiectual folosind limbajul UML (Unified Modeling Language)¹ [64] din două puncte de vedere:

- al interacțiunilor dintre obiecte prin construirea unei diagrame UML de secvențe;
- static (structural) prin construirea unei diagrame UML de clase (software) ce arată clasele și interfețele pachetului precum și relațiile dintre ele.

Pentru a construi aceste diagrame UML, am aplicat modele de proiectare Grasp generale de atribuire a responsabilitatilor claselor, precum: Information Expert, Creator, Low Coupling, High Cohesion, Pure Fabrication, Indirection [67]. Aceste modele de proiectare oferă garanția că am obținut o arhitectură de calitate ce aplică principii de proiectare de bază cum ar fi: cuplarea slabă a claselor, coeziunea înaltă a claselor, etc.

4. Referințe

- [1] S. Poslad, *Ubiquitous computing: smart devices, environments and interactions*, John Wiley & Sons, 2011. URL <http://dx.doi.org/10.1002/9780470779446>
- [2] P. Friess, *Internet of things: converging technologies for smart environments and integrated ecosystems*, River Publishers, 2013.
- [3] O.-A. Schipor, W. Wu, W.-T. Tsai, R.-D. Vatavu, *Software architecture design for spatially-indexed media in smart environments*, *Advances in Electrical and Computer Engineering* 17 (2) (2017) 17–22. URL <http://dx.doi.org/10.4316/AECE.2017.02003>
- [4] R.-D. Vatavu, C.-M. Chera, W.-T. Tsai, *Gesture profile for web services: An event-driven architecture to support gestural interfaces for smart environments*, in: *International Joint Conference on Ambient Intelligence*, Springer, 2012, pp. 161–176. URL https://doi.org/10.1007/978-3-642-34898-3_11
- [5] Y. Lou, W. Wu, R.-D. Vatavu, W.-T. Tsai, *Personalized gesture interactions for cyber-physical smart-home environments*, *Science China Information Sciences* 60 (7) (2017) 072104. URL <https://doi.org/10.1007/s11432-015-1014-7>
- [6] B.-F. Gheran, J. Vanderdonckt, R.-D. Vatavu, *Gestures for smart rings: Empirical results, insights, and design implications*, in: *Proceedings of the 2018 Designing Interactive Systems*

¹

UML este un standard descriere a modelelor create în fazele de dezvoltare a unui sistem software: analiza cerințelor software și proiectare [64]

Conference, DIS '18, ACM, New York, NY, USA, 2018, pp. 623–635. URL <http://doi.acm.org/10.1145/3196709.3196741>

[7] G. Bailly, J. Muller, M. Rohs, D. Wigdor, S. Kratz, Shoesense: A new perspective on gestural interaction and wearable applications, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12, ACM, New York, NY, USA, 2012, pp. 1239–1248. URL <http://doi.acm.org/10.1145/2207676.2208576>

[8] G. Fortino, D. Parisi, V. Pirrone, G. Di Fatta, Bodycloud: A saas approach for community body sensor networks, Future Gener. Comput. Syst. 35 (2014) 62–79. URL <http://dx.doi.org/10.1016/j.future.2013.12.015>

[9] P.-V. Cioata, R.-D. Vataavu, In tandem: Exploring interactive opportunities for dual input and output on two smartwatches, in: Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion, IUI '18 Companion, ACM, New York, NY, USA, 2018, pp. 60:1–60:2. URL <http://doi.acm.org/10.1145/3180308.3180369>

[10] X. A. Chen, T. Grossman, D. J. Wigdor, G. Fitzmaurice, Duet: Exploring joint interactions on a smart phone and a smart watch, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '14, ACM, New York, NY, USA, 2014, pp. 159–168. URL <http://doi.acm.org/10.1145/2556288.2556955>

[11] A. Esteves, E. Velloso, A. Bulling, H. Gellersen, Orbits: Gaze interaction for smart watches using smooth pursuit eye movements, in: Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST '15, ACM, New York, NY, USA, 2015, pp. 457–466. URL <http://doi.acm.org/10.1145/2807442.2807499>

[12] G. Fortino, A. Guerrieri, W. Russo, C. Savaglio, Middlewares for smart objects and smart environments: overview and comparison, in: Internet of Things Based on Smart Objects, Springer, 2014, pp. 1–27. URL https://doi.org/10.1007/978-3-319-00491-4_1

[13] M. Nebeling, E. Teunissen, M. Husmann, M. C. Norrie, Xdkinect: Development framework for cross-device interaction using kinect, in: Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '14, ACM, New York, NY, USA, 2014, pp. 65–74. URL <http://doi.acm.org/10.1145/2607023.2607024>

[14] C. Roda, A. Rodríguez, E. Navarro, V. Lopez-Jaquero, P. Gonzalez, Towards an architecture for a scalable and collaborative Aml environment, in: Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection, Springer, 2016, pp. 311–323. URL https://doi.org/10.1007/978-3-319-40159-1_26

[15] R. D. Hill, Event-response systems: A technique for specifying multi-threaded dialogues, SIGCHI Bull. 18 (4) (1986) 241–248. URL <http://doi.acm.org/10.1145/1165387.275637>

- [16] ISO, lec 25000 software and system engineering—software product quality requirements and evaluation (square)—guide to square, International Organization for Standardization. URL <https://www.iso.org/standard/35683.html>
- [17] F. Schneider, B. Berenbach, A literature survey on international standards for systems requirements engineering, *Procedia Computer Science* 16 (2013) 796–805. URL <https://doi.org/10.1016/j.procs.2013.01.083>
- [18] E. Yourdon, L. L. Constantine, *Structured design: Fundamentals of a discipline of computer program and systems design*, Prentice-Hall, Inc., 1979.
- [19] C. Moxey, M. Edwards, O. Etzion, M. Ibrahim, S. Iyer, H. Lalanne, M. Monze, M. Peters, Y. Rabinovich, G. Sharon, et al., A conceptual model for event process-ing systems, IBM Redguide publication doi:10.1.1.454.8442.
- [20] K. M. Chandy, *Event-driven applications: Costs, benefits and design approaches*, Gartner Application Integration and Web Services Summit 2006.
- [21] E. Patti, A. Acquaviva, M. Jahn, F. Pramudianto, R. Tomasi, D. Roubourdin, J. Virgone, E. Macii, Event-driven user-centric middleware for energy-efficient buildings and public spaces, *IEEE Systems Journal* 10 (3) (2016) 1137–1146. URL <https://www.doi.org/10.1109/JSYST.2014.2302750>
- [22] M. Weiser, R. Gold, J. S. Brown, The origins of ubiquitous computing research at parc in the late 1980s, *IBM Syst. J.* 38 (4) (1999) 693–696. URL <http://dx.doi.org/10.1147/sj.384.0693>
- [23] E. Aarts, R. Harwig, M. Schuurmans, *The invisible future: the seamless integration of technology into everyday life*, McGraw-Hill, Inc., New York, NY, USA, 2002, Ch. Ambient Intelligence, pp. 235–250. URL <http://dl.acm.org/citation.cfm?id=504949.504964>
- [24] M. Kuniavsky, *Smart Things: Ubiquitous computing user experience design*, Morgan Kaufmann, Burlington, MA, USA, 2010. URL <http://dx.doi.org/10.1016/C2009-0-20057-2>
- [25] D. Cook, S. K. Das, *Smart Environments: Technology, Protocols and Applications*, Wiley-Interscience, 2004. URL <http://dx.doi.org/10.1002/047168659X>
- [26] D. Korzun, I. Galov, A. Kashevnik, S. Balandin, Virtual shared workspace for smart spaces and m3-based case study, in: *Open Innovations Association FRUCT, Proceedings of 15th Conference of, IEEE, 2014*, pp. 60–68. URL <https://doi.org/10.1109/FRUCT.2014.6872437>
- [27] R.-D. Vatavu, A. Mossel, C. Schonauer, Digital vibrons: Understanding users’ perceptions of interacting with invisible, zero-weight matter, in: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI ’16, ACM, New York, NY, USA, 2016*, pp. 217– URL <http://doi.acm.org/10.1145/2935334.2935364226>.

- [28] M. A. Rahman, M. S. Hossain, A gesture-based smart home-oriented health monitoring service for people with physical impairments, in: Proceedings of the 14th International Conference on Inclusive Smart Cities and Digital Health - Volume 9677, ICOST 2016, Springer-Verlag, Berlin, Heidelberg, 2016, pp. 464–476. URL https://doi.org/10.1007/978-3-319-39601-9_42
- [29] M. J. Deen, Information and communications technologies for elderly ubiquitous healthcare in a smart home, Personal Ubiquitous Comput. 19 (3-4) (2015) 573–599. URL <http://dx.doi.org/10.1007/s00779-015-0856-x>
- [30] M.-K. Le, H.-T. Chang, Y.-M. Chang, Y.-H. Hu, H.-T. Chen, An efficient multilevel healthy cloud system using spark for smart clothes, in: Computer Symposium (ICS), 2016 International, IEEE, 2016, pp. 182–186. URL <https://doi.org/10.1109/ICS.2016.0044>
- [31] A. Pfister, A. M. West, S. Bronner, J. A. Noah, Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis, Journal of medical engineering & technology 38 (5) (2014) 274–280. URL <https://doi.org/10.3109/03091902.2014.909540>
- [32] M. Lui, J. D. Slotta, Immersive simulations for smart classrooms: exploring evolutionary concepts in secondary science, Technology, Pedagogy and Education 23 (1) (2014) 57–80. URL <https://doi.org/10.1080/1475939X.2013.838452>
- [33] B. Brown, K. O'Hara, T. Kindberg, A. Williams, Crowd computer interaction, in: CHI '09 Extended Abstracts on Human Factors in Computing Systems, CHI EA '09, ACM, New York, NY, USA, 2009, pp. 4755–4758. URL <http://doi.acm.org/10.1145/1520340.1520733>
- [34] J.-Y. L. Lawson, J. Vanderdonckt, R.-D. Vatavu, Mass-computer interaction for thousands of users and beyond, in: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, CHI EA '18, ACM, New York, NY, USA, 2018, pp. LBW032:1–LBW032:6. URL <http://doi.acm.org/10.1145/3170427.3188465>
- [35] V. G. Motti, J. Vanderdonckt, A computational framework for context-aware adaptation of user interfaces, in: Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on, IEEE, 2013, pp. 1–12. URL <https://doi.org/10.1109/RCIS.2013.6577709>
- [36] J. Vanderdonckt, Accessing guidelines information with sierraProc. of IFIP Conf. on Human-Computer Interaction, Vol. 95, 2016, 311–316. URL https://doi.org/10.1007/978-1-5041-2896-4_52
- [37] D. Korzun, On the smart spaces approach to semantic-driven design of service-oriented information systems, in: International Baltic Conference on Databases and Information Systems, Springer, 2016, pp. 181–195. URL https://doi.org/10.1007/978-3-319-40180-5_13

- [38] O. A. Schipor, et al., Improving computer assisted speech therapy through speech based emotion recognition, in: Conference proceedings of eLearning and Software for Education (eLSE), Editura Univ. Carol I, 2014, pp. 101–104. URL <http://arxiv.org/abs/1405.7796>
- [39] O.-I. Gherman, O.-A. Schipor, B.-F. Gheran, Verge: A system for collecting voice, eye gaze, gesture, and eeg data for experimental studies, in: Proceedings of the 14th International Conference on Development and Application Systems, DAS '18, 2018, pp. 150–155. URL <http://dx.doi.org/10.1109/DAAS.2018.8396088>
- [40] K. Bahreini, R. Nadolski, W. Westera, Towards multimodal emotion recognition in e-learning environments, Interactive Learning Environments 24 (3) (2016) 590–605. URL <https://doi.org/10.1080/10494820.2014.908927>
- [41] C. Ye, Y. Xia, Y. Sun, S. Wang, H. Yan, R. Mehmood, Erar: An event-driven approach for real-time activity recognition, in: Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015 International Conference on, IEEE, 2015, pp. 288–293. URL <https://doi.org/10.1109/IIKI.2015.69>
- [42] D. J. Cook, N. C. Krishnan, P. Rashidi, Activity discovery and activity recognition: A new partnership, IEEE transactions on cybernetics 43 (3) (2013) 820–828. URL <https://doi.org/10.1109/TSMCB.2012.2216873>
- [43] S. S. Rautaray, A. Agrawal, Vision based hand gesture recognition for human computer interaction: A survey, Artif. Intell. Rev. 43 (1) (2015) 1–54. URL <http://dx.doi.org/10.1007/s10462-012-9356-9>
- [44] M. R. Abid, E. M. Petriu, E. Amjadian, Dynamic sign language recognition for smart home interactive application using stochastic linear formal grammar, IEEE Transactions on Instrumentation and Measurement 64 (3) (2015) 596–605. URL <https://doi.org/10.1109/TIM.2014.2351331>
- [45] O. Schipor, S. Pentiuc, M. Schipor, The utilization of feedback and emotion recognition in computer based speech therapy system, Elektronika ir Elektrotechnika 109 (3) (2011) 101–104. URL <http://dx.doi.org/10.5755/j01.eee.109.3.181>
- [46] J. K. Zao, T. T. Gan, C. K. You, S. J. R. M'endez, C. E. Chung, Y. Te Wang, T. Mullen, T. P. Jung, Augmented brain computer interaction based on fog computing and linked data, in: Intelligent Environments (IE), 2014 International Conference on, IEEE, 2014, pp. 374–377. URL <https://doi.org/10.1109/IE.2014.54>
- [47] R.-D. Vatavu, Smart-pockets: Body-deictic gestures for fast access to personal data during ambient interactions, International Journal of HumanComputer Studies 103 (2017) 1–21. URL <http://dx.doi.org/10.1016/j.ijhcs.2017.01.005>

- [48] M. Funk, A. Sahami, N. Henze, A. Schmidt, Using a touch-sensitive wristband for text entry on smart watches, in: CHI '14 Extended Abstracts on Human Factors in Computing Systems, CHI EA '14, ACM, New York, NY, USA, 2014, pp. 2305–2310. URL <http://doi.acm.org/10.1145/2559206.2581143>
- [49] W.-H. Chen, Blowatch: Blowable and hands-free interaction for smartwatches, in: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '15, ACM, New York, NY, USA, 2015, pp. 103–108. URL <http://doi.acm.org/10.1145/2702613.2726961>
- [50] C. Goumopoulos, A. Kameas, P. Hellas, Smart objects as components of ubi-comp applications, International Journal of Multimedia and Ubiquitous Engineering 4 (3). URL http://www.sersc.org/journals/IJMUE/vol4_no3_2009/1.pdf
- [51] F. Aiello, G. Fortino, R. Gravina, A. Guerrieri, A java-based agent platform for programming wireless sensor networks, Comput. J. 54 (3) (2011) 439–454. URL <http://dx.doi.org/10.1093/comjnl/bxq019>
- [52] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi, Spine: A domain-specific framework for rapid prototyping of wbsn applications, Softw. Pract. Exper. 41 (3) (2011) 237–265. URL <https://doi.org/10.1002/spe.998>
- [53] G. Fortino, A. Guerrieri, G. M. P. O'Hare, A. Ruzzelli, A flexible building management framework based on wireless sensor and actuator networks, J. Netw. Comput. Appl. 35 (6) (2012) 1934–1952. URL <http://dx.doi.org/10.1016/j.jnca.2012.07.016>
- [54] T. Zahariadis, A. Papadakis, F. Alvarez, J. Gonzalez, F. Lopez, F. Facca, Y. Al-Hazmi, Fiware lab: managing resources and services in a cloud federation supporting future internet applications, in: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, IEEE, 2014, pp. 792–799. URL <https://doi.org/10.1109/UCC.2014.129>
- [55] F. Lab, Fiware lab (2018). URL <http://status.lab.fiware.org/>
- [56] E. Hall, The Hidden Dimension (Anchor Books a Doubleday Anchor Book), Anchor, 1966. URL <http://www.worldcat.org/isbn/0385084765>
- [57] N. Marquardt, R. Diaz-Marino, S. Boring, S. Greenberg, The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies, in: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, ACM, New York, NY, USA, 2011, pp. 315–326. URL <http://doi.acm.org/10.1145/2047196.2047238>
- [58] A. P. Volpentesta, A framework for human interaction with mobiquitous services in a smart environment, Comput. Hum. Behav. 50 (C) (2015) 177–185. URL <http://dx.doi.org/10.1016/j.chb.2015.04.003>

- [59] D. Ledo, S. Greenberg, N. Marquardt, S. Boring, Proxemic-aware controls: De- signing remote controls for ubiquitous computing ecologies, in: Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile De-vices and Services, MobileHCI '15, ACM, New York, NY, USA, 2015, pp. 187–198. URL <http://doi.acm.org/10.1145/2785830.2785871>
- [60] I. Mocanu, O. A. Schipor, A serious game for improving elderly mobility based on user emotional state, in: The International Scientific Conference eLearning and Software for Education, Vol. 2, " Carol I" National Defence University, 2017, p. 487. URL <https://doi.org/10.12753/2066-026x-17-154>
- [61] D. Chatzopoulos, C. Bermejo, Z. Huang, P. Hui, Mobile augmented reality survey: From where we are to where we go, IEEE Access 5 (2017) 6917–6950. URL <https://doi.org/10.1109/ACCESS.2017.2698164>
- [62] V. Wang, F. Salim, P. Moskovits, The definitive guide to HTML5 WebSocket, Vol. 1, Springer, 2013. doi:10.1007/978-1-4302-4741-8. URL <https://doi.org/10.1007/978-1-4302-4741-8>
- [63] D. Skvorc, M. Horvat, S. Srbljic, Performance evaluation of websocket protocol for implementation of full-duplex web streams, in: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on, IEEE, 2014, pp. 1003–1008. URL <https://doi.org/10.1109/MIPRO.2014.6859715>
- [64] Site UML specification: <https://www.omg.org/spec/UML/About-UML/>
- [65] Site Astah IDE: <http://astah.net/>
- [66] Grand, M., 2002. Patterns in Java: A Catalog of Reusable Design Patterns Illustrated with UML. (2nd ed.) John Wiley & Sons, Inc. New York
- [67] Larman, C., 2004. Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design and the Unified Process, Prentice Hall
- [68] Site Android studio: <https://developer.android.com/studio/>