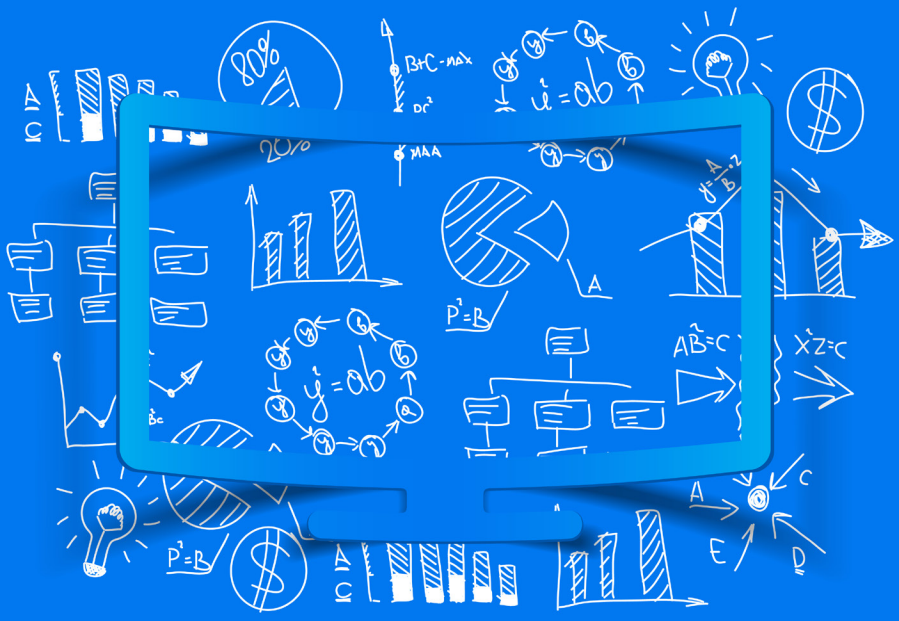
Cum să construiești un produs IT



Șerban Țîr
Sergiu Damian
Silviu Dumitrescu
Zsolt Polgar
Victor Ionescu
Voicu Oprean
Andreea Pârvu
Sebastian Big
Diana Ciorba
Monica Soare
Dan Suciu
Silvia Răusanu
Mircea Vădan
Șerban Meza
Alexandru Bolboacă
Peter Lawrey
Simona Bonghez
Bogdan Rus
Zornitsa Tomova

Cluj-Napoca, 2014

Today Software Magazine

Cum să construiești un produs IT

Această carte a fost lansată în cadrul evenimentului Cluj IT Days 2014 și va fi disponibilă *online* gratuit pe *site*-ul www.itdays.ro.

Un proiect de promovare a mediului IT românesc și a valorilor acestuia.

Cum să construiești un produs IT

Șerban Țîr
Peter Lawrey
Silvia Răusanu
Sebastian Big
Zsolt Polgar
Zornitsa Tomova
Alexandru Bolboacă
Voicu Oprean
Sergiu Damian
Simona Bonghez
Mircea Vădan
Diana Ciorba
Victor Ionescu
Dan Suciu
Andreea Pârvu
Silviu Dumitrescu
Bogdan Rus
Șerban Meza
Monica Soare

TSM TODAY SOFTWARE MAGAZINE

Redactor: Ovidiu Mățan
Coordonator: Ovidiu Mățan
Coperta: Ovidiu Mățan
Copyright: Emilia Mățan
Tehnoredactor: Ovidiu Mățan
Corector: Emilia Mățan
Traducător: Roxana Micu

Tipărit la Antoma Advertisment

ISBN 978-973-0-17970-5

Produs de
Today Software Solutions SRL

str. Plopilor, nr. 75/77
Cluj-Napoca, Cluj, Romania
contact@todaysoftmag.com

www.todaysoftmag.ro
www.todaysoftmag.com
www.itdays.ro

CUPRINS

Cuvânt înainte

IT-ul românesc a cunoscut o evoluție constantă în ultimii cincisprezece ani. Am început cu atragerea proiectelor de *outsourcing* și dezvoltarea companiilor locale. Succesul a apărut repede, iar Clujul s-a remarcat printr-un număr mare de specialiști bine pregătiți grație celor două universități locale. *Outsourcing*-ul s-a transformat gradual, trecând de la un sistem în care *project management*-ul era exclusiv străin la un mai mare *ownership* pe proiecte și la o delegare a întregilor responsabilități de dezvoltare a produselor la nivel local. Numărul mare de specialiști de top pe care îi are Clujul demonstrează o creștere consistentă a nivelului de pregătire științifică. Această evoluție rapidă precum și o oarecare nesiguranță în viitorul *outsorcing*-ului a generat pe piața românească și în particular pe cea clujeană o nevoie acută de creare a unor produse locale. În momentul de față, majoritatea companiilor experimentează diverse soluții și variante de acțiune menite să transforme aplicațiile proprii în produse locale de succes. Este un demers dificil, la a cărui reușită își dorește să contribuie și cartea de față. De altfel, titlul ales, *Cum se construiește un produs IT*, transmite tocmai această intenție de a oferi o sursă de inspirație pentru companiile locale, pentru tinerii antreprenori și programatori. Miza acestei cărți este de a reda o imagine cât mai completă asupra componentelor esențiale din ecosistemul antreprenorial: **experiența** acumulată în dezvoltarea de produse, **inovația** reprezentată prin proiectele de cercetare și **execuția** prin secțiunea tehnică și cea de management. Apariția acestei lucrări este o completare a conferinței Cluj IT Days 2014, itdays.ro, acordată participanților ca o ocazie de aprofundare a subiectelor abordate și

ca un impuls pentru noi și inedite subiecte în alte viitoare ediții. De altfel, majoritatea celor prezenți pe scenă sunt vechi colaboratori ai revistei noastre, Today Software Magazine[1].

Totodată, cartea va fi disponibilă *online* gratuit, dând astfel posibilitatea și celor care nu au fost prezenți la eveniment să descopere o parte dintre subiecte și să îi ajute în dezvoltarea viitoarelor produse de succes.

Dar, înainte de toate, avem nevoie de curajul de a realiza lucruri noi. Și pentru acestea contăm pe tine, cititorule! Dacă în această ediție în secțiunea de *leadership* se vorbește despre cum să tranziționăm de la *outsourcing* la produs, sperăm ca în ediția din 2015 a Cluj IT Days să vorbim mai mult despre produse dezvoltate local.

Mult succes în dezvoltarea de produse IT noi !!!

Ovidiu Mățan

1 www.todaysoftmag.ro și www.todaysoftmag.com

Voicu Oprean

Voicu Oprean este un antreprenor concentrat pe dezvoltarea şi distribuirea soluţiilor *software* pentru *web*, *mobile*, încorporate şi *cloud*, cu o experienţă vastă în *outsourcing* pentru industria de turism şi ospitalitate. El a înfiinţat AROBS Transilvania Software în iunie 1998, cu obiectivul principal de a promova specialiştii *software* IT români şi de a le oferi oportunităţi frumoase de carieră în România. AROBS Transilvania Software are în prezent peste 350 de specialişti angajaţi; compania are birouri operaţionale în Cluj-Napoca, Bucureşti, Iaşi, Târgu Mureş, Timişoara, Chişinău şi Londra.

1

CLUJUL - SILICON VALLEY

IT-UL: SIMPLE CIFRE SAU PUTEREA DE A SCHIMBA COMUNITĂȚI?

Clujul a devenit Silicon Valley-ul Europei, unde, conform estimărilor publicației *Paris Match*, aproximativ 20.000 de specialiști în IT dezvoltă produse menite să-i dea omului libertatea, siguranța și confortul de care are nevoie. Dacă până nu de mult piața locală de IT era bazată preponderent pe *outsourcing*, adică pe servicii prestate pentru clienții din străinătate, acum inginerii *software* clujeni nu doar că le vin în întâmpinare clienților cu soluții și produse menite să le crească businessurile, dar încep să se alinieze misiunii generale de a inova în beneficiul societății, contribuind la dezvoltatea unei industrii vitale pentru toate sferele vieții.

Datorită prezenței facultăților cu profil informatic și migrației specialiștilor din alte orașe ale țării, piața de IT din Cluj dispune de o resursă umană prețioasă, dar care este dominată de o concurență acerbă, în care companiile se zbat să atragă pe cei mai talentați oameni.

Însă provocarea reală nu ar trebui să fie cursa pentru cei mai buni oameni, indiferent de costuri, ci evoluția industriei și implicit a mediului în care trăim. Acumularea cantitativă a determinat saltul calitativ în IT, iar acum companiilor le revine misiunea de a se concentra pe dezvoltarea unor produse inovatoare. Aceste produse care ar putea să se impună la nivel global sunt cele care pe termen lung vor face diferența.

În definitiv, rolul nostru, al IT-știlor este de a mișca lucrurile, de a optimiza procese, de a reduce costuri și a îmbunătăți în egală măsură activitățile cotidiene ale oamenilor de rând și ale mediului de afaceri.

EVOLUȚIA. DE LA OUTSOURCING LA PRODUS

La fel ca alte companii de profil, AROBS și-a făcut intrarea pe piață cu servicii de *outsourcing*, consolidându-și poziția datorită credibilității acumulate în timp și competenței oamenilor implicați. Dar nu ne-am oprit aici, ci am trecut la un alt nivel odată cu dezvoltarea unor produse cu valoare adăugată mare, care au întărit brandul și au adus beneficii palpabile utilizatorilor.

În 2003, am lansat *Optimall SFA*, un produs de automatizare a vânzărilor, care ajută în momentul de față peste 3.000 de agenți de vânzari și de mercantizori să își optimizeze activitatea și să-și crească controlul în relația cu clienții lor din diverse zone ale țării.

În 2006, am lansat soluția de monitorizare a parcului auto *TrackGPS*, utilizată de peste 2.000 de clienți, pe mai mult de 30.000 de mașini. A mai fost și soluția pentru automatele de bilete de transport în comun. Un întreg municipiu – Iași, capitala Moldovei – folosește acest produs și sunt convins că acesta este doar începutul.

Manualele digitale pentru copiii din clasele I-II-a sunt un alt produs de care suntem foarte mândri, pentru că avem convingerea că, în acest fel, putem contribui la educația viitoarelor generații și la dezvoltarea învățământului românesc, aflat în picaj în ultimii ani.

Pe lângă produsele de mai sus, pe care le considerăm vârfuri de aisberg, AROBS are și alte produse incubate, care vor fi lansate în lunile ce urmează. Toate sunt rezultatul a sute de ore de creație, agitație și nesomn, dar și răspunsul la micile eșecuri cărora a trebuit să le facem față de-a lungul timpului. Pentru că nu orice inovație devine automat vitală pentru utilizatori, a trebuit și noi să trăim pe pielea noastră experiența dezvoltării multor produse care nu au reușit să intre în topul preferințelor consumatorilor. După ce am creat o soluție pentru agențiile de turism, care a convins doar zece clienți și încă una pentru gestionarea charterelor de către tur-operatori, a trebuit să acceptăm acest rezultat sub așteptări și să ne orientăm spre alte produse, prin care să țintim direct nevoile beneficiarilor.

FOCUS AND STRATEGY DETERMINE THE OUTCOME

In perfect harmony

There are passionate people at AROBS.
You fit in naturally.

AROBS
Transilvania Software

At AROBS we believe in doing things the right way. We create a harmonious work and life balance, constantly striving to develop ourselves. In everything we do, we respect the environment, the nature and the people, we celebrate life.

With us you'll find support, openness, and warmth.

job@arobs.com

Connect with us on

Încă de la început, visam la o companie completă, de aceea după *outsourcing* și soluțiile *software*, pasul firesc era să dezvoltăm produse *hardware*. Așa, am început fabricarea în masă, în China, a produselor sub brandul *Smailo*. Acesta include primul sistem de navigație din România- un brand 100% românesc, tablete Android, camere auto DVR și *actioncamera*, foarte prețuite de amatorii de sporturi extreme de la scufundări până la săriturile cu parapanta.

Noi, la AROBS ne ghidăm după o filosofie pe cât de simplă pe atât de benefică: încurajarea inovației și a spiritului antreprenorial. Pornind de aici, în ultimii ani, ne-am axat pe creșterea calității și a cantității produselor pe care le livrăm în tandem cu dezvoltarea capacităților angajaților, prin implicarea lor în proiecte internaționale. Experiența acumulată în acest fel o transpunem ulterior în proiectele interne, fapt ce favorizează inovația și lansarea pe piață a unor produse create de oameni pentru oameni. Fără îndoială, o greutate mare în procesul de creație îl are și *background*-ul lor informațional și cultural – din echipele noastre fac parte astăzi specialiști români, maghiari, olandezi, coreeni, irlandezi sau basarabeni -, astfel că produsele scoase pe piață de AROBS le dau un sentiment de diversitate și universalitate celor

care le aleg.

Trăim într-un oraș în care IT-ul a devenit noua „industrie grea" a secolului. Clujul ne-a pus la dispoziție resursele sale, iar acum e timpul ca noi să ne punem în slujba comunității, oferindu-i, în schimb, produse și servicii de înaltă calitate, performante și inovatoare.

AROBS și-a asumat deja acest rol, dar voi? Ce-ar fi dacă, în 2015, fiecare companie ar lansa cel puțin un produs cu valoare adăugată? Orice drum începe cu primul pas, spune un vechi proverb. Așa că de ce nu ar face fiecare dintre noi acel pas ca să schimbăm fața comunității?

Doar în acest fel lansând produse inovative, Clujul va deveni un adevărat Silicon Valley!

Dacă i-am urma sfatul lui Tony Hsieh, fondatorul Zappos - „*Distrați-vă! Jocul e mult mai savuros atunci când încerci să faci și altceva, nu doar bani*" -, am construi piesă cu piesă o lume mai bună. În definitiv, ceea ce lăsăm după noi este mai presus de orice strategii de moment.

Autor

Voicu Oprean – CEO @ Arobs

Șerban Țîr

Şerban Ţîr are o experienţă de 14 ani în lucrul cu companii având baza în Silicon Valley. În această perioadă a jucat roluri diferite în cadrul proiectelor *software* în care a fost implicat, începând de la dezvoltator *software* până la Vice Preşedinte în Inginerie. Experienţa sa include relaţii de afaceri atât cu startup-uri mici, cât şi cu companii publice foarte mari. În prezent, este CTO al grupului de companii Gemini Solutions.

El a înţeles valoarea reală a inovaţiei tehnologiei de vârf şi este unul dintre iniţiatorii Gemini Foundry, un „accelerator flexibil" care promovează start-up-urile *high-tech* româneşti pe piaţa americană.

2

TIPARE DE GÂNDIRE ÎN ANTREPRENORIAT

O INCURSIUNE ÎN MEDIUL ANTREPRENORIAL DIN ROMÂNIA

ECOSISTEMUL ANTREPRENORIATULUI ROMÂNESC

În România, antreprenoriatul în sine presupune o gândire disruptivă dacă luăm în calcul istoria ultimilor ani. Înainte de revoluția din 1989 nu se putea vorbi de antreprenoriat pe teritoriul țării noastre, iar în cei 25 de ani ce s-au scurs de atunci multe dintre tentativele antreprenoriale *high tech* au falimentat rapid fie din cauza dezinteresului general în a susține astfel de inițiative, fie din cauza puterii reduse de cumpărare sau de a investi .

Dacă dorim să ne concentrăm pe domeniul IT, numărăm pe degetele de la o mână antreprenorii ce au pornit din România și au înregistrat un succes pe plan global. În plus, salariile din mediul IT oferă celor care le primesc o plasă de siguranță care face trecerea către antreprenoriat cu atât mai grea și netentantă. Antreprenoriatul în sine poate fi asemănat cu un salt dificil la trapez fără plasă de siguranță: cine reușește, culege toți laurii, iar cine nu …

De aceea, considerăm că antreprenoriatul în România presupune o gândire disruptivă, o ieșire din tipare, un "NU" ferm spus vieții de salariat și o dorință acerbă de a îți lua destinul în propriile mâini. În ultimii ani însă observăm că acest curent începe să prindă în rândul tinerilor care sunt din ce în ce mai dornici să învețe cum pot deveni proprii stăpâni. Lor le dedicăm acest articol.

Presupunem că sunteți un tânăr antreprenor în IT, eventual parte a unei echipe ce are o idee în care crede cu tărie și sunteți gata de a renunța la co-

moditatea vieții de angajat într-o corporație, pentru a-și vedea visul realizat: **La ce ar trebui să vă gândiți? Ce întrebări ar trebui să vă puneți?**

Mai jos încercăm să trecem prin câteva dintre ele. Întâmplător sau nu, în mare coincid cu punctele pe care un eventual investitor (fie el instituțional sau privat, VC sau *business angel*) vrea să le audă pentru a lua în calcul o investiție. Ele constituie scheletul a ceea ce se numește „*pitch* către un investitor".

Pentru a face lucrurile mai ușor de înțeles, ne propunem să condimentăm explicațiile tehnice cu exemple de "așa nu", parte dintre ele "gustate" în practică.

Acest articol nu se vrea unul critic la adresa mediului antreprenorial românesc, care a evoluat mult în ultimii ani. Greșelile subliniate sunt destul de des întâlnite și pot fi evitate ușor cu puțină atenție. Se vrea așadar un îndrumar/ajutor venit în întâmpinarea celor care-și construiesc *pitch*-ul și se pregătesc să atragă o investiție.

SUBIECTE LEGATE DE "BUSINESS MODEL"

Modelul de business este un termen amplu care descrie modul în care o organizație creează și livrează o plus-valoare. Acesta definește aspectele cheie ale unui business, precum scop, infrastructură, structuri organizaționale, procese utilizate, public țintă și multe altele. Însă, dacă anumite puncte ale acestui model pot fi dezvoltate într-o etapă ulterioară, fiecare antreprenor ar trebui să trateze de la început cu seriozitate următoarele aspecte:

Valoarea adăugată (value proposition) – constă într-o descriere simplă, scurtă și ușor de înțeles a ceea ce este businessul / serviciul pe care îl oferi: Ce

oferi, ce nevoie adresezi și de ce ar cumpăra cineva de la tine?

Diferențiatori cheie (key differentiators) - Ce te deosebește de ceilalți de pe piață, de concurență ? În literatura de specialitate avem teoretic de ales între câteva variante, bine delimitate. Sau poate, în cazul cel mai complex, între combinații ale lor.

"Unfair competitive advantages" – oricât m-aș strădui mi-e greu să traduc acest termen. Poate: „avantaje ce provin dintr-un start furat". Mai simplu spus: ce ași ai în mână pentru ca altcineva să nu facă același lucru ca tine. Fie mai repede, fie mai bine. De ce o corporație care are resurse mult superioare ție, nu ți-ar lua-o înainte, eventual cu un produs mai bun.

Dacă, în general, primele două subiecte sunt puse în evidență, cel de-al treilea pare tabu pentru antreprenorii autohtoni. În rare cazuri am auzit pomenindu-se fie și câteva cuvinte despre el.

NDA

Vom prezenta ceea este NDA și în ce constă semnarea unui NDA. Am auzit de atâtea ori "Am o idee grozavă, ți-o voi spune după ce semnăm un NDA ca să fiu sigur că nu mi-o furați". Să furăm ce? O idee? Atenție! NDA-ul este menit să protejeze un business în sine cu specificul și diferențiatorii lui și nu o idee sau o nișă de piață. Ca să nu mai adaug că investitorii serioși nu semnează NDA.

ANALIZA DE PIAȚĂ

TAM VS. SAM VS. SOM

Evident că orice antreprenor trebuie să-și înțeleagă piața în cele mai mici detalii. În plus față de aceasta, care este o pre-condiție a succesului unei inițiative antreprenoriale, în cazul în care antreprenorul vrea să obțină finanțare, acesta trebuie să știe cărei piețe se adresează. Realitatea arată însă destul de des confuzii legate de piața totală adresabilă și de segmentul pe care speră antreprenorul să-l obțină din această piață.

Pentru o mai bună înțelegere a acestor noțiuni recomandăm citirea următorului articol explicativ: https://www.thebusinessplanshop.com/blog/en/entry/tam_sam_som

COMPETIȚIE

Se pare că în România, afirmația "Noi nu avem competiție reală" este foarte frecvent asimilată unei stări de fapt care continuă să persiste în mediul antreprenorial. Fiți foarte reticenți în a spune acest lucru! Dacă nu ați avut cumva o idee absolut genială ("una la un miliard") atunci acest fel de aserțiune ascunde o insuficientă cunoaștere a pieței.

Iată mai jos capitolele care ar trebui atinse ca parte a analizei competiției:

- Oferă altcineva o soluție la problema pe care o rezolvi?
- Sunt mulți competitori sau câțiva jucători care controlează piața?
- Ce oferi tu diferit de aceștia? Cu ce ești mai bun față de ceilalți?
- Dacă o altă companie a încercat să rezolve aceeași problemă, dar a eșuat. Care au fost cauzele acestui eșec? Ce s-a întâmplat? Unde a greșit? De ce a eșuat?

POTENȚIAL DE CREȘTERE ȘI SCALABILITATE

Este foarte important de înțeles că pentru investitorii în *high-tech*, potențialul de creștere al businessului primează în fața unei surse imediate de venit. Ei își asumă faptul că investițiile în startup-uri din zona IT sunt investiții de risc ridicat. Tocmai de aceea țin foarte mult ca în cazul unei reușite aceasta să fie de magnitudine ridicată. În plus, este mult mai ușor să crești profitul obținut decât să atragi utilizatori noi în aplicație sau să îi menții pe cei deja existenți.

Deși mulți dintre antreprenori știu, subliniem și noi că pentru investitori este mult mai important ca afacerea să aibă un potențial de creștere, scalabilitate mare, decât obținerea unui *break-even* rapid / venituri imediate.

CLIENȚI

În general un startup rezolvă o problemă care îi este familiară fondatorului său. Lovindu-se deja de acea problemă, fondatorul poate da o direcție inițială în speranța că alte persoane, care s-au lovit de aceeași problemă, i se vor alătura în utilizarea produsului / serviciilor oferite. Din acest moment este

foarte important ca *startup*-ul să asculte *feedback*-ul clienților pentru a găsi o direcție de dezvoltare favorabilă și a nu cădea în capcana: "Eu am problema asta, doar eu știu ce e mai bine să facă acest produs". În modul acesta clienții se întorc într-o aplicație pe care o găsesc pe placul lor și pot atrage după ei alți potențiali utilizatori.

De aceea, când vorbim de clienți este foarte important să avem în vedere următorii indicatori:

STRATEGII DE ACHIZIȚIE ȘI COSTURI

Care vor fi primii pași în atragerea utilizatorilor? Vorbești cu prietenii, utilizezi rețelele sociale, forumuri, etc.? Aceste strategii trebuie puse în balanță în funcție de costurile angrenate (timp/bani) și numărul potențial de clienți pe care îl poți atrage.

Un topic care lipsește foarte des din analiza antreprenorilor locali se referă la achiziția a ceea ce se numește "early adopters" - adică utilizatori timpurii ai produsului. Auzim prea des expresii de genul *"după ce voi avea zece mii de utilizatori, se va propaga viral"*. În lipsa unei strategii adecvate de atragere a utilizatorilor timpurii, este posibil să nu ajungeți nici măcar acolo. Utilizatorii timpurii sunt de regulă cei mai greu de atras.

TRACȚIUNE ȘI RETENȚIE

Tracțiunea este un indicator foarte important și în principiu reprezintă "dovada" că piața este interesată de ceea ce construiești tu, deoarece are nevoie de acel produs. Este important să aduci mereu utilizatori noi și să-i transformi în clienți fideli. Altfel, toate investițiile pe această parte vor fi în zadar. Vei avea într-adevăr foarte mulți utilizatori, dar care nu se mai întorc a doua zi.

Cel mai adesea "aceasta nevoie a pieței" se măsoară la modul concret pentru *site*-urile *web* de exemplu în utilizatori unici, număr de utilizatori noi în comparație cu utilizatorii care au părăsit platforma, număr de utilizatori activi etc. . Subliniem însă că tracțiunea (împreună cu diversele mărimi care o pot măsura) nu este singurul mod în care se poate evalua valoarea unui business.

PROPRIETATE INTELECTUALĂ ȘI PATENTE

Un exemplu simplu de înțeles este cel legat de un produs care are părți, algoritmi, idei patentabile. Cu cât numărul de posibile patente este mai

mare cu atât evaluarea crește. De asemenea crește și încrederea investitorului că produsul / serviciul nu poate fi copiat.

MONETIZARE

Am menționat anterior că potențialul de creștere primează în fața obținerii unor venituri imediate. Acest lucru însă nu înseamnă că monetizarea nu trebuie să vă mai preocupe deloc. Este indicat să aveți un plan de a obține niște venituri de pe urma produsului / serviciilor oferite, chiar dacă acest lucru se va întâmpla după ceva timp.

ECHIPA

Echipa este sufletul fiecărui startup, totodată și principalul factor luat în considerare de un potențial investitor atunci când acesta hotărăște să investească. Aceasta se datorează faptului că după ce va face investiția se presupune că va petrece suficient timp cu membrii echipei sau cel puțin cu liderul ei. Trebuie să se simtă relaxat și încrezător pentru a petrece acest timp. Vom detalia în rândurile următoare atuurile unei echipe.

DE CE NOI?

- Ce anume diferențiază această echipă de altele?
- De cât timp lucrați împreună?
- Proiectele trecute dezvoltate împreună cu echipa sunt un bonus - chiar dacă ele nu au mers ideal. Orice eșec este o lecție învățată.
- Echipa nu va mai trebui să treacă de obstacolele pe care le întâmpină în perioada de formare când fiecare își găsește un rol și învață să lucreze cu ceilalți membri.

EXPERIENȚE COMPLEMENTARE

Echipa ideală este compusă din trei membri cu experiențe complementare: o persoană tehnică, o persoană cu experiență de business și una în marketing, eventual design. Fiecare persoană trebuie să aibă rolurile foarte bine definite și să își cunoască responsabilitățile. Aceasta nu înseamnă că echipe formate din două sau patru persoane nu pot reuși într-un startup. Dar contează ca membrii echipei să fie experți pe arii diferite.

TOATĂ LUMEA ARE CUNOȘTINȚE

Trăim într-o lume în care chiar contează "pe cine cunoști". Deși meritocrația nu caracterizează încă foarte mult societatea noastră, acest lucru nu trebuie să ne sperie. Perseverând vom putea crea condițiile instaurării unei meritocrații. Cu cât veți înțelege mai repede "regulile jocului" cu atât le veți putea folosi mai repede în avantajul vostru. Duceți-vă la conferințe, faceți *networking* și creșteți-vă cercul de cunoștințe.

Puneți-vă în locul unei persoane care întâlnește zilnic zeci de echipe, fiecare cu câte o idee genială și o promisiune de a schimba lumea. Cum faci diferența între cei care vând iluzii și cei care chiar pot duce visul până la stadiul de realitate? Începi să favorizezi echipele care vin cu referințe de la oameni în care ai încredere. Dacă una sau mai multe persoane din cercul tău de cunoștințe recomandă pe cineva, privești cu alți ochi acea echipă.

SINDROMUL ANTREPRENORULUI CU JUMĂTATE DE NORMĂ

Unul dintre cele mai des întâlnite scenarii este următorul: "Am un proiect la care lucrez în timpul liber. Două ore pe zi și uneori în weekend." Aceasta poate fi o soluție pe termen scurt, mai ales în ceea ce privește primii pași, însă pe termen lung este doar o cale sigură către eșec. Un startup nu ar trebui să fie acel lucru pe care îl faci când găsești puțin timp liber sau când vii deja obosit după o zi de muncă. Pentru a avea succes va trebui să fii implicat 100% în startup-ul tău deoarece obstacolele pe care le vei întâlni vor avea nevoie de toată atenția ta, iar la sfârșitul zilei aceasta va face diferența între cei care reușesc și cei care rămân în urmă.

Cu acest articol ne reafirmăm ideea că mediul antreprenorial românesc a evoluat mult în ultima perioadă și este în continuă evoluție. O problemă este încă lipsa de pe piața autohtonă a caselor de "*venture capitalists*", dar și în acest caz credem că este doar o chestiune de timp.

Odată acest ecosistem format din antreprenori-incubatoare-investitori, nu va mai fi mult până vom vedea startup-uri românești de real succes pe piețele globale. Trebuie doar răbdare.

Autori

Șerban Țîr – CTO, Gemini Solutions
Radu Popovici - Associate, Gemini Solutions Foundry

Alexandru Bolboacă

Sunt un dezvoltator de *software* pasionat, o persoană care învaţă continuu, mentor şi profesor la Mozaic Works. Sunt un programator poliglot, iubesc provocările şi utilizez toate tehnicile pe care le cunosc pentru a le depăşi. De anul trecut, dezvolt şi produse.

Câteva informaţii despre mine:

Am facilitat primul Code Retreat în afara Statelor Unite (cu Maria Diaconu) şi am ajutat la definirea formatului.

- 15 ani de dezvoltare software
- 5 ani de C++
- 7 ani de C#
- 5 ani de predare, mentorat şi instruire
- Iubesc arhitectura software, designul, programarea. În prezent învăţ cum să creez produse extraordinare.
- Am ajutat la crearea comunităţilor AgileWorks în şase oraşe mari din România.

Nu ezitaţi să mă contactaţi pentru sfaturi, programare în pereche (chiar şi la distanţă) sau oportunităţi de dezvoltare în acest domeniu, pe website-ul meu http://alexbolboaca.ro sau prin Mozaic Works http://mozaicworks.com.

3

O METODĂ STRUCTURATĂ DE LUCRU PENTRU STARTUP-URI

Dezvoltarea unui produs implică mult efort, înțelegere și adaptare la realitățile pieței. Datorită modificărilor multiple, presupunerea antreprenorilor este că abordarea „fă orice e posibil" e suficientă. Deși dezvoltarea unui produs este o activitate care se desfășoară la limita haosului, ea poate fi realizată într-un mod controlat și ordonat.

Iată un model simplificat:

- Clarificarea unui scop și a unei viziuni.
- Validarea viziunii prin experimente.
- Iterat de la 1 (fie clarificarea și mai detaliată, fie pivotare).
- Pentru validarea viziunii, următoarele unelte pot fi folosite:Identificarea MVP-ului folosind Personas și Storymapping,
- Planificarea implementării folosind *user stories,*
- Identificarea presupunerilor și a experimentelor necesare,
- Folosirea Kanban pentru o abordare disciplinată a dezvoltării,
- Utilizarea ATDD și a *mockup*-urilor pentru a elimina cât mai mult din necunoscut.

Fiecare dintre tehnicile menționate mai sus este detaliată în rândurile următoare.

SCOP ȘI VIZIUNE

Totul pornește de la un scop și de la o viziune asupra produsului. Nu voi insista mult pe aceste subiecte, pentru că s-au scris nenumărate articole și cărți despre ele. Merită totuși reamintite câteva aspecte cheie.

O viziune bine definită ajută la luarea rapidă a deciziilor în timpul dezvoltării produsului și a businessului. Deși ne-am dori să luăm toate deciziile într-un mod bazat pe date, realitatea este că rareori un business poate aștepta momentul în care toate datele sunt clare și corect interpretate. Aproape orice decizie de business se ia pe baza unor informații parțiale. În lipsa informațiilor complete, alte elemente contează pentru luarea deciziilor: intuiția echipei și a antreprenorului și mai ales viziunea produsului.

Viziunea „do no evil" a Google este clar definită în filozofia companiei[1]:

- Nu permitem ca anunţurile publicitare să fie afişate pe paginile noastre de rezultate, decât dacă sunt relevante acolo unde apar.
- Noi credem că publicitatea poate să fie eficientă fără a fi ţipătoare.
- Publicitatea pe Google este întotdeauna marcată clar drept „Link sponsorizat", astfel încât să nu compromită integritatea rezultatului căutării noastre.

Scopul produsului este diferit de viziune. Dacă viziunea se referă la filozofia produsului, scopul este măsurabil. Acesta poate fi legat de: număr de utilizatori, venituri, număr de vizite, etc. .

Lucrez în acest moment la un produs eHealth. Viziunea lui este să ajute actul medical al medicilor de familie pentru a preveni greșelile și a simplifica partea administrativă. Scopul pe care ne concentrăm în acest moment este să reducem timpul petrecut de doctor în medie cu introducerea datelor pacienților. Viziunea este greu de măsurat, scopul îl putem măsura prin compararea cu alte aplicații similare.

TEHNICI UTILE

PERSONAS

Noțiunea de persona provine din comunitatea UX. O persona nu este altceva decât un profil de utilizator, care oferă contextul necesar pentru experimente și alegerea funcționalităților. Profilul de utilizator poate fi îmbogățit

1 https://www.google.com/about/company/philosophy/

incremental.

Nume: Dan[2]
Specializare: Doctor de Familie
Vârstă: 36 ani
Familie: Necăsătorit
Alte detalii:

- *deține o tabletă și un iPhone*
- *are cont pe Facebook*
- *își face concediile în străinătate*

Personas pentru un produs pot fi identificate de echipă printr-un workshop de câteva ore.

E important de înțeles că și *personas* implică presupuneri legate de produs și de utilizatori. E bine ca aceste presupuneri să fie pe cât posibil validate, fie prin statistici demografice fie prin interviuri cu potențiali utilizatori.

STORYMAPPING

Storymapping este o tehnică de a identifica pașii necesari pentru îndeplinirea unei activități, din punctul de vedere al utilizatorului. Fiecare pas poate fi îndeplinit în mai multe moduri fie în aplicație, fie în afara ei.

Exemplu de *storymap* pentru prescrierea unui medicament:
Selectare medicament -> Stabilire doză -> Indicații administrare -> Imprimare prescriere

Text liber
Căutare medicament
Căutare subst. activă
Căutare clasă medicamente

E bine ca echipa să pornească de la identificarea scenariului complet, urmând ca apoi să selecteze ce anume va fi implementat și ce anume pare a oferi cel mai mare beneficiu pentru utilizator. Cu cât sunt implementați mai repede pașii care aduc valoare pentru utilizatori, cu atât cresc șansele de conversie rapidă.

2 Sursa imaginii: http://pixabay.com/p-217140/

USER STORIES

User story-ul este cea mai răspândită modalitate de gestionare a cerințelor din Agile (XP, Scrum) și Lean/Kanban. Avantajul unui *user story* față de documentele de cerințe este că permite rafinarea incrementală a cerințelor pe măsură ce echipa înțelege mai bine utilizatorii și piața.

User story-ul este și un element de dialog între utilizatori, programatori, designeri și testeri. În cea mai simplă formă a sa, un *user story* arată astfel:

„Ca un <rol> vreau să <acțiune> astfel încât <beneficiu>"

De exemplu:

„Ca medic de familie vreau să găsesc un medicament pe bază de nume astfel încât să îl pot prescrie rapid"

User story-ul este însă foarte adaptabil și permite modificări sau adăugarea unor elemente de care au nevoie fiecare din cei implicați în produs. De exemplu:

- Un *mockup* va permite programatorilor să înceapă rapid implementarea și va asigura integritatea designului grafic.
- Teste de acceptanță vor permite validarea completă a *user story*-ului și implementarea sa corectă.
- Informații suplimentare pentru *deployment*, în cazul în care există probleme specifice.

În cazul utilizării personas, *user story*-ul se poate referi la o persoană în loc de un rol. Acest lucru ajută la ajustarea sa în funcție de profilul utilizator.

Cum se schimbă user story-ul de selectare medicament pentru a converti utilizatorul următor?

Nume: Grigore[3]
Specializare: Doctor de Familie
Vârstă: 53 ani
Familie: Căsătorit, doi copii
Alte detalii:

- *puțin familiarizat cu tehnologia*
- *scrie la tastatură foarte încet*
- *are nevoie de ochelari pentru citit*

3 http://whatisstomachflu.com/wp-content/uploads/Senior-medical-doctor-Large.jpg

I T.A.K.E. Unconference 2015

28-29 May 2015, Bucharest

The only **technology agnostic unconference** in Central and Eastern Europe! Participants will find out new **concepts and ideas,** will **learn new techniques** and enjoy the company of practitioners from all over Europe at the **3rd I T.A.K.E Unconference edition** - http://itakeunconf.com

The schedule includes sessions on **Architecture, DevOps, Product Development, Technical Leadership or Hardcore Programming**. During OpenSpace you get the chance to be a speaker yourself and to include in the program the topics you are interested in.

HOW IS IT SPECIAL?

Top 4 reasons: 1) It's technology agnostic, so you will learn no matter what you code in at work. 2) You get to meet leading software craftsman from Europe and USA. 3) Talks are only 25% of the event, the rest is hands-on work. 4) It's small enough that you can talk to virtually everybody.

WHO WILL BENEFIT THE MOST?

The main audience: **Craftsmanship Programmers, Architects, DevOps, Technical Leaders, CTOs and Managers, Technical Co-Founder, Technical Consultant**.

FIRST CONFIRMED KEYNOTES

Simon Brown is an award-winning speaker and author of Software Architecture for Developers - a guide to software architecture, technical leadership and the balance with agility.

James Lewis introduced evolutionary architecture practices and agile software development techniques to various blue chip companies.

REGISTRATION

Registration is now open at http://2015.itakeunconf.com/register

For any questions: alexandra.marinescu@mozaicworks.com

MOCKUPS

Fiecare *user story* care implică interfață utilizator are nevoie de un *mockup* pentru a fluidiza dezvoltarea. *Mockup*-ul poate fi creat în diferite moduri: desenat pe foaie, într-un *tool* de design precum Adobe Photoshop, în *tool*-uri specializate de *prototyping* sau direct în *html.*

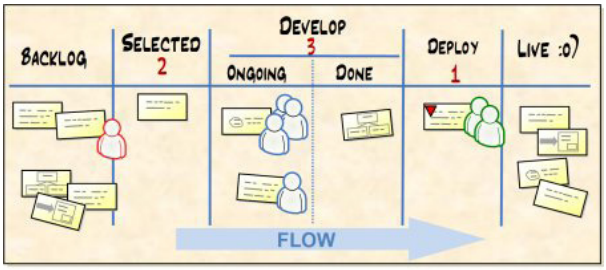
Mockup-urile sunt cel mai bine create prin colaborarea între designeri, programatori și persoana care a elaborat viziunea produsului. Abordarea interdisciplinară permite identificarea mai multor detalii și clarificări la momentul potrivit.

KANBAN

Cea mai potrivită tehnică de organizare a dezvoltării pentru un startup este Kanban. Kanban introduce foarte puține reguli, dar destule pentru a permite o dezvoltare fluentă și îmbunătățire continuă.

Pașii pentru a adopta Kanban sunt următorii:

1. Identificarea etapelor din procesul de dezvoltare:
    - Procesul de dezvoltare este reprezentat de etapele : Analiză, Dezvoltare, Testare, Deployment, Done.
    - Pot exista însă multe variații.
    - Procesul trebuie să fie cât mai aproape de realitate la momentul respectiv.
    - Procesul se va modifica în timp, deci nu investiți prea mult timp în identificarea exactă a lui.
    - În cazul unei echipe noi, cel mai simplu proces este și cel mai bun pentru a începe rapid.
2. Construirea unui *board* conform cu etapele din proces



3. Adăugarea la boara *user story*-urilor și *task*-urilor ce trebuie făcute și scopul lor.
4. Afișarea *board*-ului într-un loc foarte vizibil pentru întreaga echipă.

Kanban implică două aspecte fundamentale:

- **Disciplină în execuție**: concentrarea pe un singur scenariu la un moment dat, pe unul-două *stories* (în cazul unei echipe mici) și pe finalizarea muncii în progres („work in progress") la calitatea necesară.
- **Îmbunătățirea continuă a procesului**: atunci când ceva nu merge bine (apar blocaje, *story*-urile se întorc dintr-o coloană din dreapta spre stânga, apar probleme de calitate), echipa trebuie să decidă ce să modifice pentru a preveni apariția aceluiași tip de problemă.

Kanban necesită pe lângă cele discutate și:

- limitarea muncii în progres,
- adăugarea de criterii de intrare pe fiecare coloană, pentru a avea o calitate predictibilă,
- monitorizarea *cycle time* și *lead time* pentru a permite îmbunătățirea predictibilității și reducerea *time to market.*

Aceste aspecte sunt detaliate în diverse resurse pe internet.

ACCEPTANCE TEST DRIVEN DEVELOPMENT

Această tehnică permite comunicarea între dezvoltatori și persoanele care se concentrează pe funcționalitățile produsului.

Pornind de la criteriile de acceptanță definite pentru un *user story*, programatorii scriu teste automate care sunt ușor de citit de către persoane non-tehnice. Pentru a le scrie, pot fi folosite tool-uri precum Fitnesse, Selenium sau librării BDD precum Cucumber. Testele nu vor trece atâta timp cât funcționalitatea nu este implementată.

Test de Acceptanță scris într-o sintaxă similară cu Cucumber
Feature: Find drug by name
In order to prescribe a drug fast
As a GP

I want to find the drug by name

Scenario: Find Aspirin by name
Given I am logged in and in the prescribe drug screen
When I search for Aspirin
Then I find all commercial names for aspirin (e.g.: Ecotrin, Acetylsalicylic Acid, Bayer Aspirin)

Scrierea acestor teste permite clarificarea mai detaliată a cerințelor, înainte de a fi nevoie de modificări costisitoare în cod.

CONCLUZIE

Deși dezvoltarea unui produs în cadrul unui startup este o activitate care se desfășoară la limita haosului, ea poate fi organizată într-un mod disciplinat și previzibil. Avantajele acestui mod de organizare sunt evidente: eliminarea pierderilor datorate problemelor de calitate și *rework*, micșorarea stresului echipei și deschiderea posibilității de inovație prin reducerea administrației necesare în condiții de haos.

Echipa Mozaic Works (www.mozaicworks.com) poate ajuta la adopția rapidă și eficientă a acestor tehnici, permițând echipelor să se concentreze asupra scopului lor principal: crearea unui produs de succes.

Autor

Alexandru Bolboacă - Trainer & Coach @ Mozaic Works

Mircea Vădan

Pasiunile mele sunt tehnologia, educația și antreprenoriatul. Primele activități antreprenoriale le-am început în studenție, prin mai multe proiecte non-profit dedicate studenților, în paralel cu frecventarea facultății (Calculatoare, UTCN).

În ultimii 3 ani, activitățile mele s-au desfășurat exclusiv în zona startup-urilor, mi se pare domeniul potrivit de a îmbina pasiunile enumerate mai sus. Acum lucrez în cadrul startup-ului ZenQ (www.zenq.co) și ajut la formarea comunității de startupuri din Cluj (www.clujstartups.com).

4

VALIDAREA PROBLEMEI ȘI A SOLUȚIEI PRIN INTERVIURI

UN PRIM PAS ÎN DEZVOLTAREA UNUI PRODUS

Acum mai bine de un an, la începutul verii 2013, în timp ce lucram la startup-ul UseTogether (platformă de schimburi de obiecte), am analizat mai atent metricele de pe platformă, cu dorința de a hotărî direcția viitoare. În acel moment existau aproximativ 2000 de utilizatori ai platformei, așa că a fost suficient pentru a extrage două concluzii importante:

- 3% dintre utilizatorii noi postau măcar un articol și doar 1% din ei adăugau mai mult de 3;
- 95% din obiectele/anunțurile postate nu a fost cerute de altcineva prin platformă.

Așadar, nu avea sens să continuăm pe aceeași direcție. De aceea, ne-am propus să schimbăm ceva. Dar ce? Ne-am gândit la schimbarea mesajului de marketing, la a ne orienta mai mult pe căutarea de obiecte și la concentrarea pe o nișă specifică, cercetând următoarele segmente de piață: studenți, părinți și angajați. Considerăm că fiecare dintre aceste categorii ar putea avea o oarecare nevoie de folosi o platformă de acest fel, am vrut să vedem care dintre ele are o nevoie mai mare.

După două săptămâni de cercetare, cea mai promițătoare părea a fi nișa părinților cu copii mici. Aceștia au nevoie de multe articole pentru copii, cheltuiesc o grămadă de bani pe ele și sunt folosite pentru o perioadă scurtă de timp și, în cele din urmă, acele produse ajung doar să ocupe spațiul sau garajul. De asemenea, important a fost faptul că obiceiul părinților este deja format, astfel încât părinții fac schimb, împrumută sau fac cadou acele arti-

cole în cercurile lor de prieteni.

Oricum, pentru a fi siguri de acest lucru, am început să facem mai multe interviuri de validare a problemei cu diverși părinți: mai tineri și mai în vârstă, cu un venit mai mic sau mai mare, cu copii mici sau cu copii de 5-7 ani. Voiam să fim cât mai siguri și să înțelegem părinții, nevoile și comportamentul lor precum și modul în care o platformă ca a noastră îi poate ajuta. Din fericire am avut unele cunoștințe dobândite în unele workshopuri privind validarea problemei și astfel aveam noțiunile de bază cu care să pornim. Pe lângă câteva chestionare și *focus* grupuri, am avut și câteva zeci de interviuri față în față, acestea fiind cele mai importante că să ne dăm seama de nuanțele problemei.

Mai jos voi descrie **structura generală aplicată la interviuri**. Mi s-a părut un model bun și destul de obiectiv. Practic, există câteva întrebări de bază, iar alte întrebări satelit pot fi adăugate în jurul acestora. Materialele studiate pe care s-a bazat structura de mai jos le găsiți la sfârșitul capitolului.

În primul rând, pentru a obține o validare sinceră, persoanele intervievate nu ar trebui să știe nimic despre ceea ce vreți să construiți (sau cu cât mai puțin cu atât mai bine). Ideea este de a pune întrebări generale și apoi de a intra mai adânc în descrierea problemei, prin întrebări deschise.

1. „Care sunt problemele pe care le întâmpini legate de ...?"

De exemplu, la întrebarea "Spune-mi despre problemele pe care le-ai întâlnit, legate de creșterea copiilor", probabil ei vor menționa problemele de sănătate, căutarea informațiilor necesare, banii cheltuiți pe diverse lucruri pentru copil. În acest pas, ar trebui să existe o referire la problema pe care doriți să o abordați. În exemplul de mai sus este vorba de cheltuieli pe diverse lucruri. Apoi pot urma alte întrebări în așa fel încât să simțim, ca intervievatori că am înțeles bine perspectiva intervievatului asupra problemei/nevoii existente.

2. „Care sunt variantele pe care le utilizezi pentru a rezolva această problemă?"

Deoarece este o problemă, oamenii încearcă să o rezolve cumva, *offline* (de exemplu: își sună prietenii, cer ajutor profesionist, merg la evenimente de vânzare-cumpărare obiecte sau vizitează magazine) sau *online* (căutare pe Google, postarea pe grupuri de Facebook, verificarea *site*-urilor specializate - OLX.ro, Tocmai.ro, etc.). Prin găsirea variantelor folosite îți dai seama care

va fi concurența ta directă sau indirectă, prin prisma celui intervievat.

3. „Ce probleme vezi la aceste variante?"

Poate că aceste variante sunt prea greu de folosit sau este nevoie de mult timp/ mulți bani sau au *user-experience* neplăcut. Aceste aspecte sunt și motive pentru a utiliza produsul dvs. comparativ cu acele alternative sau sunt sugestii despre ce ar trebui să eviți sau să aplici mai bine. Ideea este ca produsul tău să rezolve o problemă și să aducă valoare în plus clienților potențiali, comparativ cu ceea ce există deja pentru ei. Fără a lua în considerare acest aspect, nu prea mai are sens acest demers.

4. „Care este soluția perfectă pentru tine?"

În acest moment, intervievații s-au gândit deja profund la situația lor și la câteva idei despre cum lucrurile pot fi îmbunătățite, astfel încât poți să-i îndemni să-și imagineze soluția potrivită/ideală pentru ei. S-ar putea obține unele sfaturi surprinzătoare, idei creative sau într-adevăr diferite față de ceea ce aveai în minte. În cazul în care primești răspunsuri care sunt apropiate de soluția gândită de tine anterior, soluția ta este validată cel puțin prin această metodă.

Aici poate să se încheie prima parte a interviului. În acest moment poți menționa ceea ce ai de gând să faci, poți descrie detaliile, *feature*-uri și poți cere *feedback* direct, arată chiar și *wireframe*-uri. Aceasta este de fapt a doua parte a interviului, o discuție liberă cu scopul de a primi cât mai mult *feedback* legat de idee.

În final, e important să îi întrebi cu cine ar mai trebui să vorbești sau pe cine îți pot ei recomanda? Toți acești intervievați pot constitui utilizatorii inițiali, pentru că ai stabilit deja cu ei o legătură care îți poate oferi *feedback* în continuare.

Repetă acest tip de interviu de cel puțin 20 de ori pe segmentul semnificativ. Adaugă chestionare *online* și *focus* grupuri, în scopul de a obține mai multe idei sau pentru a confirma ceea ce știi deja și abia apoi începe construirea propriu-zisă a produsului.

CÂTEVA SFATURI

- Nu construi nimic fără validare cu potențiali clienți.
- Este important să afli ce funcționează și ce nu, înainte de a construi un MVP.

- Stabilește categorii ale posibililor clienți și clasifică interviurile pe diverse pe segmente
- Descoperă în fiecare segment pe cei care au potențial de *early adopters* adică pe cei pentru care produsul tău e *must-have*, nu *nice-to-have*.
- Realizează interviuri până când vei vedea că răspunsurile sunt similare sau se repetă des, pentru a extrage concluzii relevante.
- Reiterează interviurile sau adaptează-le în funcție de rezultatele din interviurile anterioare.
- Utilizează cuvinte ale clienților în frazarea mesajului de marketing.
- Atunci când vorbești despre produs, subliniază mai întâi beneficiile pentru client.
- Pe cât posibil, fă o astfel de validare pentru fiecare *feature* nou.
- Provocarea în acest moment nu este de a obține noi utilizatori, ci de a obține utilizatorii pentru care produsul rezolvă într-adevăr o problemă.
- Evidențiază fapte și întâmplări reale, care să reflecte comportamentul autentic. De multe ori utilizatorii au o părere, dar au comportament diferit față de ceea ce afirmă.

În ceea ce privește acest subiect, materialele următoare s-au dovedit a fi foarte utile:

1. Founder Centric (Rob Fitzpatrick, Sal Virami)
   - cartea "The Mom Test"[1]
   - prezentarea "Early Stage Customer Development"[2]
   - prezentarea "Practical Customer Development - Lean Startup"[3]
3. Andreas Klinger[4]
4. Steve Blank:
   - http://steveblank.com/category/customerdevelopment/
   - http://www.slideshare.net/venturehacks/customer-development-methodology-presentation
   - http://www.youtube.com/watch?v=6t0tCXPpyM .
5. Cindy Alvarez's blog[5]

Fiecare dintre autorii menționați mai sus vine cu un punct de vedere ușor

1 http://momtestbook.com
2 http://vimeo.com/40192415
3 http://www.slideshare.net/foundercentric/practical-customer-development-lean-startup-groningen
4 http://www.slideshare.net/andreasklinger/actionable-customer-development
5 http://www.cindyalvarez.com/learning/faq-customer-development-for-product-managers

diferit, dar complementând perspectiva validării problemei și a soluției. Toți ajung la aceeași concluzie: "construiește ceva ce oamenii își doresc, verifică ipotezele prin fapte și experiențe reale, vorbește cu posibili clienți cât mai mult posibil" .

Nu cred că această metodă a interviurilor este infailibilă, mai ales că în unele cazuri, când e vorba despre anumite nevoie neconștientizate încă în mintea posibililor utilizatori, este chiar greu de aplicat. În asamblu mi s-a părut un proces foarte folositor care în acel moment ne-a ajutat să înțelegem mult mai bine problema părinților și abordarea de urmat.

Efortul investit în dezvoltarea unui produs este foarte mare și riscul este pe măsură. Dar prin metodele elaborate și folosite de *startup*-uri, un produs poate ajunge de la început mai aproape de nevoile clientului. Dacă ai nevoie de ajutor în acest proces, scrie-mi un e-mail la mircea.vadan@gmail.com

Autor

Mircea Vădan - clujstartups.com

Sebastian Big

Sebastian Big, pe care îl găsiţi pe sebastianbig.ro, este unul dintre cei mai buni clarvăzători sintactici pe plan național, o meserie de viitor în timpuri de criză nu atât economică, cât conceptuală.

Profeţii semantice de orice natură: socială, psihologică, tehnologică etc., s-au încercat de multe ori în trecut, dar rezultatele au fost dezamăgitoare iar direcţiile de urmat neclare, în ciuda eforturilor asidue ale respectivelor scene de a le urma.

Interaction Designer, dar de formaţie filozofică şi cu traduceri din Virilio, Baudrillard, Derrida & Stiegler şi un doctorat în Comunicare terminat prematur, Sebastian Big îşi valorifică talentul pe piaţa românească de IT, încercând să pună în valoare metodele sale de clarviziune într-un sistem deocamdată cantonat în factualitatea imediată.

5

CUM PUTEM GÂNDI INTERACTIV *USER EXPERIENCE*-UL?

După cum precizează majoritatea încercărilor de definire a acestui domeniu multidisciplinar, UX-designul încorporează, pe de o parte, aspecte din discipline tehnice: informatică, arhitectură, design grafic, design industrial și, pe de altă parte, se străduiește să integreze aspecte din științele umane: științe cognitive, psihologie, antropologie, sociologie. De ce științe umane? Păi, e foarte simplu, pentru că au ca domeniu de activitate tocmai omul (*user*-ul) și existența (experiența) lui. Științele umane sunt însă o invenție destul de recentă (cel mult 150 de ani) față de disciplina cu același obiect de activitate, din care s-au dezvoltat și ele: filosofia. În ce privește omul și ce i se întâmplă acestuia, filosofia are o "expertiză" de peste 2000 de ani.

Istoria filosofiei se împarte îndeobște în trei perioade: antichitatea, modernitatea și contemporaneitatea, fiecare cu tematica ei specifică. Obiectul principal al filosofiei antice a fost "ceea ce este", **ființa**, modernitatea s-a ocupat de **cunoaștere**, iar tematica vremurilor noastre e **comunicarea**, **tehnologia informației** fiind un caz particular al acestei ultime etape. Fiecare dintre aceste etape e marcată conceptual de o relație opozitivă. Vom vedea pe parcursul acestei prezentări felul în care a rezolvat filosofia aceste conflicte și vom încerca să ne dăm seama în ce măsură rezolvările ei ne pot ajuta în mai buna înțelegere a conceptului de *user experience*.

CELE TREI ETAPE ALE FILOSOFIEI ȘI CONFLICTELE LOR

ANTICHITATEA GREACĂ - ESENȚĂ VS. EXISTENȚĂ

Filosofia se naște în Grecia antică, odată cu trecerea de la ritualurile credințelor primitive la discursul rațional. Absolutul credințelor primitive este abstractizat în Principii. La fel ca și în cazul sacrului, omul trebuie să le cunoască pentru a putea înțelege lumea. Marea problemă a acestei etape a apărut odată cu aprofundarea relației dintre principiu și manifestările sale. Ce este aparență și ce este realitate, ce e iluzoriu și ce există cu adevărat în această lume, schimbarea sau statornicia?

Prima soluție la această problemă o dă Heraclit din Efes. După el, universul se transformă în fiece clipă. Totul este pus în mișcare de un dinamism infinit, iar principiul e puterea schimbării, manifestată prin mișcare. „Totul curge" (*Panta rhei*), totul este (în) mișcare, tot ce există este prins într-o continuă mobilitate.

O altă soluție vine de la Parmenide din Elea, care neagă schimbarea / transformarea. Pentru el, din punctul de vedere al principiului, al Ființei, schimbarea și mișcarea sunt iluzorii, iar devenirea este doar o aparență. Numai simțurile sunt vinovate de această iluzie pentru că ne fac să credem în curgerea necontenită a fenomenelor. Rațiunea împrăștie iluzia, arătându-ne ce este real, Ființa unică, imobilă, eternă, ascunsă sub valul aparențelor multiple.

PERIOADA MODERNĂ - RAȚIONALISM VS. EMPIRISM

Perioada modernă e marcată de descoperirile geografice, de trecerea de la perspectiva geocentrică la cea heliocentrică, stând în general sub semnul schimbării modurilor de cunoaștere. Instalarea și ascensiunea cunoașterii în cultura occidentală este concomitentă cu instalarea științelor moderne, a cunoașterii științifice moderne. Problema modernității e următoarea: dacă realitatea este relativă la cunoaștere este vital să știm care este sursa cunoașterii adevărate. E real ceea ce cunoaștem prin simțuri sau ceea ce cunoaștem prin rațiune? Și aici, ca și pentru filosofia antichității, se desprind două soluții posibile.

Empirismul postulează teoria conform căreia experiența senzorială e cea

care face posibilă cunoașterea noastră. Simțurile ne dau toate datele brute despre lume, material fără de care nu ar exista cunoașterea. Percepția este punctul de plecare al unui proces la finalul căruia ajungem să avem certitudini. Doar percepția este cea care poate aduce cunoaștere.

Raționalismul pretinde că nu simțurile, ci rațiunea e punctul de plecare al oricărei cunoașteri. Raționaliștii pretind că fără categorii și principii a priori furnizate de rațiune nu am putea organiza și interpreta experiența noastră senzorială. Am avea de-a face cu un imens, nediferențiat caleidoscop de senzații fără semnificație.

Doar principiile și conceptele generate și certificate de rațiune și doar ceea ce e deductibil logic din ele poate oferi cunoaștere.

CONTEMPORANEITATE - STRUCTURĂ VS NARAȚIUNE

Nu voi descrie aici felul în care se desfășoară lucrurile în contemporaneitate. Suntem cu toții prezenți. Voi spune doar că, în ce privește filosofia, avem în continuare de-a face cu o relație conflictuală, reprezentată de această dată de școala structuralistă și de cea hermeneutică. Cele două au evident moduri diferite de a vedea comunicarea.

Structuralismul este o orientare teoretică și metodologică interdisciplinară care studiază structura, funcțiile și sistemele de relații ce caracterizează realitatea, punând în prim-plan totalitatea în raport cu individul și sincronicitatea faptelor în raport cu istoria. Semnificația e un rezultat al relațiilor interne ale sistemului ca întreg. Comunicarea ar fi în acest caz cu atât mai precisă cu cât indivizii implicați în actul ei stăpânesc structura respectivului sistem.

Pentru hermenenutică, ce pune în față individul, realitatea e compusă din împletirea unor narațiuni individuale. Înțelegerea se realizează prin interpretarea perpetuă a propriilor narațiuni și compararea lor cu narațiunile celorlalți și cu povești generice (biblia, talmudul etc.). Istoria ca desfășurare temporală are preeminență în acest caz, diacronia are câștig de cauză în fața sincroniei. În cazul hermeneuticii comunicarea e un proces nesfârșit în care semnificația e întotdeauna nouă, momentană și emoțională.

REZOLVĂRILE CELOR TREI ETAPE

ANTICHITATE

Problema cu perspectiva lui Heraclit e că ajungem să ne situăm într-un relativism total în care nu există nimic statornic, cunoaștere, morală, legalitate, sentimente etc. . Dacă îi dăm dreptate lui Parmenide, salvăm toate aceste forme și conținuturi ale vieții noastre, dar ajungem să ne situăm în contradicție cu lumea pe care ne-o prezintă simțurile noastre.

Platon e primul care leagă termenii acestei dihotomii, spunând că lumea pe care o oferă senzațiile noastre, lumea fenomenelor naturale și a corporalității, schimbătoare, trecătoare și contradictorie, e o copie a lumii ideilor, a realității absolute, eterne și adevărate, introducând astfel în istoria filosofiei diferența dintre imanent și transcendent. Cele două sunt legate, însă transcendentul poate exista și fără copia lui palidă, imanentul. Mai cu picioarele pe pământ, Aristotel ia doctrina platoniciană, abstractizează lumea ideilor, reducând-o la cauze (categorii) cu înțeles de principii, care stau la baza oricărei existențe individuale. Lucrurile și ființele există în potență, dar nu se pot actualiza fără a fi înscrise în categorii. Pentru a înțelege mai bine relația posibilitate-actualizare, să ne luăm la două exemple: copilul este posibilitatea, iar actualizarea lui este ființa umană matură; un proiect este în posibilitate, iar actualizarea lui este un obiect, o clădire, o aplicație. Nu există categorie fără actualizare, așa cum nu există nici actualizare fără categorie.

MODERNITATEA

Problema raționalismului e că aplicând doar rațiunea la experiență, e foarte probabil că vom ajunge la iluzii teoretice. Cu empirismul problema e că experiența ar fi total subiectivă și haotică fără să fie filtrată de rațiune.

Kant e cel care împacă cele două curente de gândire - categoriile sunt cele care sintetizează datele diversității senzoriale în obiecte inteligibile. Simțurile procură minții intuițiile, iar capacitatea de judecată produce judecăți asupra acestor intuiții, grupându-le în categorii. Categoriile ridică intuițiile din conștiinta momentană a subiectului, plasându-le într-o conștiință generală, producând conținuturi de cunoaștere universale. Astfel constituit, subiectul este capabil să organizeze influențele haotice pe care le exercită asupra sa mediul său și să constituie prin această organizare obiectul cunoașterii sale.

CONTEMPORAN

Problema cu structuralismul e că semanticul vine într-un plan secundar în raport cu sintacticul. Semnificația e secundară gramaticii. Comunicarea e pur funcțională și golită de emoție. Riscul abordării hermeneutice, pe de altă parte, e acela al unei rețele de comunicare fără cap și coadă, de tip Turn Babel, imposibil de gestionat, în care comunicarea reală ar fi *random*.

Rezolvarea conflictului dintre cele două perspective e ceva ce se întâmplă *as we speak*. Structura este cea care ordonează diversitatea narațiunilor personale. De la comunități complet izolate în spațiu și timp am ajuns în *satul global*, cu limbaj și *pattern*-uri de comportament globale. Gândim (structurat) global și acționăm (producem narațiuni) local.

CONCLUZII

După cum am văzut, fiecare dintre cele trei perioade din istoria filosofiei e marcată de două feluri de a vedea lucrurile, la prima vedere diametral opuse, fără putință de împăcare. Fiecare perspectivă are însă lipsurile ei, niciuna dintre ele nu reușește să ofere o viziune globală, funcțională asupra epocii respective. De fiecare dată, lucrurile încep să capete sens în momentul în care apare cineva care să le pună în interacțiune. Deși nu e o muncă ușoară conceptual, simplul fapt de a vedea posibilitatea interacțiunii dintre cele două perspective deschide un front de lucru care nu poate decât să ducă la rezolvarea situației opozitive.

Afirmam mai spus că epoca în care trăim e marcată de tehnologie. Dacă pentru antici era vorba despre a trăi, pentru moderni despre a cunoaște, iar pentru contemporanul secolului trecut despre a comunica, în prezent, legătura sau felul în care oamenii interacționează cu tehnologia poartă numele de *user experience*. Să vedem, în încheiere, în ce măsură ne putem folosi de *know-how*-ul filosofiei pentru a defini cuplul opozitiv care marchează *user experience*-ul și pentru a gestiona această relație aparent dihotomică.

Dacă ar fi să analizăm *user experience*-ul printr-un ochi-rezumat al celor trei etape, avem pe de o parte un nucleu dur, care spune că numai principiile, categoriile și structura contează, că tehnologia este ceva ideal, care nu are nevoie de nicio poveste, de nicio zorzoană care să distragă atenția de la adevărul ei transcendent, la care *user*-ul nu poate ajunge decât printr-un travaliu de aceeași intensitate cu cel depus de nucleul dur de mai sus.

La capătul opus se află cei înzestrați cu harul *soft-skill*-urilor, din perspectiva cărora principiile și categoriile sunt ceva de ocolit, structura e ceva

colateral, iar ceea ce contează cu adevărat e doar impresia artistică care să gâdile simțurile *user*-ului. Interfața noastră cu tehnologia e un talmeș-balmeș nesustenabil, generat de un *user research* menit să satisfacă apăsările artistice și de comunicare ale departamentului de marketing.

Cum am văzut din povestea filosofiei, niciuna dintre variante nu rezolvă singură problema *user-experience*-ului. Un *user-experience* strict bazat pe principii, categorii, structuri e gol, rece și greu de folosit, tocmai pentru că nu ține cont de *user*-i. Un *user-experience* bazat doar pe simțuri, percepții și narațiuni e haotic și în cele din urmă la fel de greu de folosit, neavând o structură.

Ce se întâmplă în momentul în care punem să interacționeze cele două perspective? Structura adună și abstractizează narațiunile, transformându-le în *pattern*-uri. Narațiunile sunt cele care furnizează cărămizile din care e construită structura, care nu e altceva decât un sistem de componente și comportamente, adică un *framework*. Parafrazând un celebru om politic român: "Existența unui framework nu se discută, se afirmă."

Nu e o întâmplare deci faptul UX-ul funcționează, așa cum spuneam în introducere, la intersecția dintre disciplinele tehnice, reci, formale, logice și științele umane, care știu să înțeleagă și să pună în scenă latura emoțională a *user*-ului nostru cel de toate zilele.

Autor

Sebastian Big - interaction designer @ Betfair

Diana Ciorba

Diana Ciorba coordonează cu dedicare strategia de marketing a companiei Codespring, din 2009. Furnizând constant informații despre piață și despre expertiza echipei, ea a devenit un ambasador activ al comunității IT din Cluj.

Sărbătorind 11 ani de provocări în marketing și dezvoltarea afacerilor, de-a lungul ultimilor 7 ani a avut și rolul de consultant independent pentru investitori din țară și străinătate. Alături de experți și manageri executivi, pentru ea, munca multi și interdisciplinară este o plăcere.

Înțelegând antreprenoriatul ca un stil de viață diferit, creșterea afacerilor este întotdeauna o preocupare prioritară. Diana consideră activitatea de marketing și dezvoltare în ITC un teren propice pentru o minte jucăușă.

Studiile din Franța au influențat parcursul academic al Dianei, ea deținând în prezent un master în Management. Cu un palmares ce include premii în domeniul artelor dramatice, scrierii creative și științifice, Diana este un autentic specialist hibrid.

6

PRIVIND CU OCHII UTILIZATORULUI FINAL

Construirea unui produs *software* viabil este un proces complex, de durată, care implică specialiști din domenii diverse. Adeseori la întrebarea *CUM ?* răspunsul cel mai sincer pe care îl pot da este: *Depinde!* Astfel, semnalizez în mintea interlocutorului multitudinea de factori ce afectează modalitatea în care se poate ajunge la rezultatul dorit. Implicit, păşim pe terenul întrebărilor exploratorii – întrebări esenţiale în demersul decizional pe care îl parcurgem în vederea construirii unui produs *software* şi nu numai (!). Cu cât vom avea răspunsuri mai clare, mai cuprinzătoare, mai documentate, cu atât decizia va fi mai bine fundamentată şi mai rafinată.

Din experienţa directă, am reţinut că atât echipele tehnice, cât şi cele comerciale care participă la construirea produselor *software* şi la punerea la punct a proceselor corelate, cad de acord asupra unui aspect fundamental: *construim pentru oameni şi aceasta schimbă tot*.

#1 CONSTRUIM PENTRU OAMENI

Pentru a ne da seama de complexitatea domeniului trebuie doar să observăm cele două elemente implicate. De o parte regăsim utilizatorul uman; de cealaltă parte se află computerul. Comunicarea dintre cele două elemente se face cu un scop prestabilit, în funcţie de domeniu. Tipul de *input* acceptat de computer şi tipul de *output* pe care programul îl furnizează va determina interfaţa cu utilizatorul.

Oamenii au nivele diferite de pregătire şi cunoştinţe în utilizarea com-

puterelor și dispozitivelor ce încorporează tehnologie înaltă. Avem de-a face cu o gamă largă de utilizatori – de la *utilizatori elementari* până la *utilizatori extremi*.

Echipa de dezvoltare a unui nou produs sau a unei noi versiuni a unui produs deja existent va trebui să țină cont de această diversitate și să hotărască cea mai eficientă abordare. Totodată, va trebui să ținem cont de faptul că utilizatorul final și cumpărătorul sunt două specii diferite. Produsul *software* construit trebuie să fie bine primit de *utilizatorul final.*

#2 OAMENII VOR SĂ SE SIMTĂ BINE

Din naștere, omul caută experiențele care îl fac să se simtă bine. Fie că avem de-a face cu un program destinat activităților profesionale, fie celor recreative, utilizatorul final va trebui să aibă o experiență plăcută în interacțiunea cu produsul *software* și cu dispozitivul pe care este instalat.

Experiența plăcută a utilizatorului unui produs *software* este greu de definit datorită factorilor cu un grad mare de complexitate care o afectează. În funcție de domeniul de aplicabilitate și tipul de utilizator, stabilirea unui set de criterii prin care se va evalua nivelul de satisfacție în utilizare va spori șansele dezirabilității produsului în piață.

#3 OAMENII CAUTĂ SĂ FIE LIBERI

Printre așteptările față de tehnologia de vârf se numără și obținerea unui grad sporit de libertate, precum: libertatea de a se mișca, libertatea de a explora (mediul înconjurător, datele, fenomenele).

Acest fapt a determinat și permanenta evoluție a tehnologiilor de interacțiune între om și computer de la cele non-perceptuale (tastatură, joystick, mănușă electronică, stilet, dispozitive și suprafețe tangibile) spre cele perceptuale (de vedere, auz și percepție la distanță). În plus, există deja tehnologii ce permit *input further out:* cele bazate pe gesturi, palpație, biometrie sau impulsul nervos.

În funcție de domeniul de aplicabilitate al produsului *software*, echipa de proiectare va trebui să analizeze specificul activității utilizatorilor finali pentru a putea oferi un produs performant, de încredere și care să elibereze corpul de manevrarea unor obiecte și de prezența într-o anumită locație.

Având aceste principii fundamentale în minte, demersul construirii unui produs *software* poate avea două direcții: a) cineva are o idee ce va trebui testată și verificată sau b) se pornește analiza nevoilor nesatisfăcute de către produsele existente în scopul soluționării acestora. În ambele cazuri, buna practică presupune analiza cerințelor de piață și conturarea cerințelor pe care produsul va trebui să le îndeplinească.

#4 MRD (MARKET REQUIREMENT DOCUMENT)

Documentul care va articula noul produs și planul de lansare al acestuia este MRD (Market Requirement Document), uneori numit și PRD (Product Requirement Document) sau BRD (Business Requirement Document). Conținutul său vizează identificarea problemei de rezolvat sau nevoii de acoperit, analiza cât mai detaliată și cuprinzătoare a acesteia, propunerea de soluții, dezvoltarea și testarea.

Acesta va avea în centrul atenției utilizatorul și modul în care acesta va interacționa cu produsul nou lansat sau cu ultima sa versiune. Adeseori s-ar putea ca utilizatorul să găsească noi variante de utilizare, la care proiectanții nici nu s-au gândit. Tocmai de aceea activitățile de prototipare și testare cu potențiali utilizatori finali trebuie prevăzute în planul de dezvoltare.

Prin acest document se vor deschide și opțiunile decizionale legate de afacerea proprie:

- Putem identifica un concept pe care potențialii clienți să dorească să îl probeze?
- Putem identifica o strategie de marketing eficientă pentru a introduce și creste produsul pe piață?
- Va corespunde produsul condițiilor noastre de profitabilitate și rentabilitate?

Dacă răspunsurile la întrebările de mai sus vor fi pozitive, se poate trece la predarea cerințelor și ideilor deja generate către echipa de dezvoltare care va parcurge procesul de dezvoltare al unui produs *software*.

#5 REGLAJUL FIN LA FIECARE PAS

Atât echipa responsabilă de aspectele comerciale cât și cea de aspectele

tehnice vor fi cu atât mai performante cu cât vor fi mai receptive. Pentru a crește receptivitatea de ansamblu, coordonatorul activității de marketing și cel al dezvoltării produsului vor trebui să pună la punct un sistem prin care să existe în permanență puncte de contact pentru preluarea informației de la piața țintă, de la clientul ordonator, de la utilizatorii finali.

Receptivitatea are ca scop final creșterea șanselor de a oferi din start o valoare autentică în fiecare etapă de dezvoltare a produsului *software*. Valoarea percepută de utilizatorii finali și cumpărători va determina strategia de preț a produsului și capacitatea de a câștiga piața.

Un exemplu în sensul creșterii valorii percepute per ansamblu este programul pus la punct la Codespring: **reglajul fin la fiecare pas.**

Figura 1: Procesul de dezvoltare software bazat pe principiul reglajului fin la Codespring / sursa: Catalogul Codespring, ed. 2014

Fondatorul companiei, ing. Levente Szelyes alături de echipa sa, au dezvoltat parteneriate strategice pentru dezvoltarea de produse *software* complexe cu câțiva din liderii mondiali în industriile lor. În paralel, compania are și produse *software* proprii. Rezultatele pozitive au fost posibile datorită acurateții cu care se derulează procesele de analiză, planificare și dezvoltare.

Acuratețea perceptibilă în produsele *software* dezvoltate de Codespring are la baza acest mecanism sofisticat de reglaj fin al tuturor variabilelor și resurselor implicate în proces.

Mai mult, **reglajul fin** a devenit un principiu guvernator al organizației, în relație cu piața, clienții existenți și cei potențiali. Mecanismul stă la baza câștigării încrederii clienților și menținerii flexibilității organizației pe măsură ce se dezvoltă.

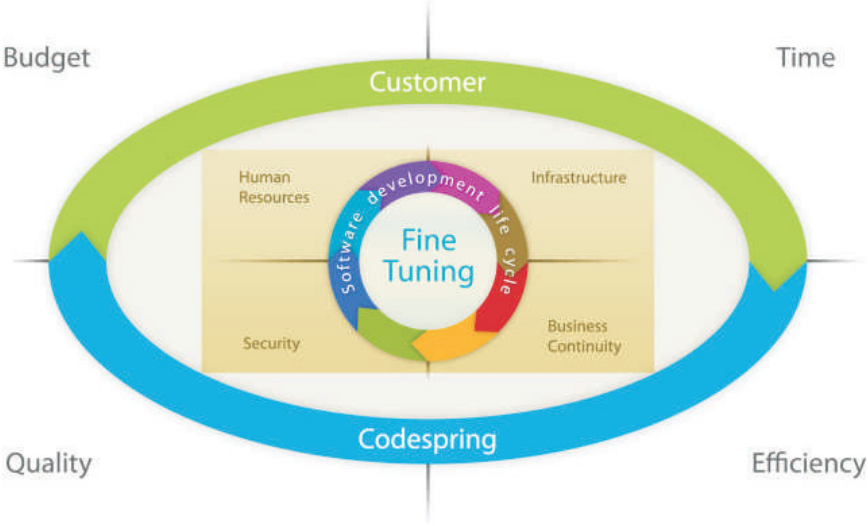


Figura 2: Reglajul fin ca principiu guvernator al organizației / sursa: Catalogul Codespring, ed. 2014

#6 IMPLICAREA UTILIZATORILOR ÎN PROTOTIPARE ȘI TESTARE

Timpul de proiectare și lansare pe piață a unui produs *software* de calitate poate varia în funcție de resursele disponibile: tehnice, financiare și umane. Practica din domeniu a consacrat prototiparea și testarea ca două etape ce pot avea o influență majoră asupra calității rezultatului final. Din perspectiva utilizatorului, acestea sunt momente critice în care se poate manifesta și poate participa la conturarea produsului pe care probabil îl va folosi în viitor.

Observarea utilizatorilor în interacțiunea cu prototipurile vor oferi informații valoroase în legătură cu: reacția față de probabila interfață, modul în care ar utiliza funcționalitățile produsului, experiența pe care ar putea-o trăi utilizatorul. Ulterior, echipa de proiectare va putea propune schimbări,

îmbunătățiri și completări. Avantajul colaborării în această etapă este apropierea de un rezultat cât mai util și plăcut utilizatorului țintă.

Testarea cu utilizatorii țintă este o evaluare concluzivă a produsului înainte de distribuirea pe piața țintă. În cazul unei evaluări pozitive, se poate considera lansarea pe piață și creșterea produsului. Este o etapă prin care se pot colecta informații despre comportamentul utilizatorilor și despre viteza cu care adoptă noul produs sau noua versiune.

#7 PRIETENIA CU NOILE GENERAȚII DE UTILIZATORI

Dacă până acum vorbeam de noua generație de tehnologie, e timpul să atragem atenția și asupra noilor generații de utilizatori. Viteza cu care absorbim fiecare nouă dezvoltare tehnologică și cea cu care copiii și adolescenții reușesc să le exploateze, indică o creștere vertiginoasă a abilității acestora de a interacționa cu tehnologia. Comportamentul acestor noi generații va diferi, fiind deja influențat de hiperconectivitate, mobilitate și accesul la realități diferite.

Apropierea organizațiilor ITC de noile generații este imperativă. Consider că persoanele având rol de strateg în dezvoltarea afacerilor, respectiv în dezvoltarea de produse *software* - în special, pot găsi o sursă de inspirație pentru viitorul apropiat analizând comportamentele proeminente din rândul noilor generații. Adeseori, sinceritatea brută a copiilor și adolescenților poate ajuta dezvoltatorii unui nou produs în identificarea punctelor slabe sau unor noi oportunități. Bineînțeles și pentru obținerea acestor informații o sistematizare a mijloacelor de colectare este necesară.

#8 INOVAREA CA STARE MENTALĂ

Având în minte utilizatorul final, construirea unui produs *software* trebuie să aibă în vedere o îmbunătățire semnificativă sau chiar o premieră a modului de rezolvare a unei probleme date. Prin caracteristicile intrinseci ale domeniului ITC, dezvoltarea produselor *software* este un proces creativ în relație imediată cu inovarea.

Inovarea de produs, de proces, de organizare, de metodă sau de marketing are loc atunci când condițiile sunt prielnice. Terenul pentru inovare la scară

mare se poate pregăti din timp prin adresarea unor întrebari simple – de genul: „Cum am putea opera un pacient la distanță?" sau „Cum am putea învăța tipul de lovituri pe un teren de golf, fără instructor?" sau „Cum am putea monitoriza sănătatea ecosistemului marin?"... întrebările se vor adresa atât specialiștilor cât și potențialilor utilizatori finali. Adresând întrebări și așteptând răspunsuri antrenăm echipa să vină cu idei, soluții și inducem utilizatorilor finali ideea că o soluție ar fi posibilă pentru nevoia lor.

Aliniind organizația și beneficiarii direcți la ideea de propunerea a unor soluții, creăm podul spre inovare.

#9 UTILIZATOR FINAL NU ÎNSEAMNĂ CUMPĂRĂTOR

Un produs *software* poate fi construit pentru uz direct, personal (pe computerul personal, pe telefonul mobil personal, etc.) sau pentru uz direct, profesional (pe computerul sau stația de lucru a companiei, pe telefonul de serviciu, în mașina de serviciu, etc.). Deși în ambele cazuri, echipa de proiectare va avea în vedere utilizatorul final, va trebui făcută distincția între situația în care acesta are și puterea de a decide achiziția și situația în care echipa de achiziții a companiei va decide pentru el.

Lanțul decizional va afecta resursele alocate dezvoltării produsului *software*, versiunea finală și modalitatea de punere pe piață.

#10 COMUNICAREA UMANĂ ESTE ESENȚIALĂ

Prezența sau lipsa unui salut cordial, verbal sau în scris, poate decide startul oricărei colaborări. Pentru a asigura un nivel ridicat de veridicitate a datelor colectate despre utilizatori și nevoile lor, un volum și o varietate adecvată de date pentru conceperea sau dezvoltarea produsului *software*, comunicarea trebuie să fie bine pusă la punct: ce întrebări se vor adresa, cum vor fi transmise, cum se vor primi răspunsurile, în ce context (locație, număr de persoane prezente, timpul, durata, etc.).

Ulterior, comunicarea rezultatelor către echipa de proiect și modul în care acestea vor fi înțelese vor contura forma și structura primară a produsului. Pe parcursul dezvoltării comunicarea interpersonală și de grup (de echipă) va influența major viteza de rezolvare a sarcinilor, calitatea soluțiilor și fiabili-

tatea lor. În final, comunicarea produsului și a beneficiilor sale către publicul țintă va determina succesul comercial al produsului dezvoltat.

Privită din perspective diverse, construirea unui produs software poate fi descrisă diferit. Definirea corectă a *utilității* acestuia ne duce spre utilizatorul final, adică *oamenii* care îl vor folosi în activitățile lor. Astfel *utilizatorul final* este punctul de convergență al tuturor specialiștilor implicați în acest proces. Privind prin *ochii utilizatorului final* știm către ce trebuie să ne îndreptăm și tot de aici pornim pentru a reevalua și a demara următoarea etapă.

BIBLIOGRAFIE

1. Catalog Codespring, ediția 2014.
2. Diana Ciorba - Interviu: „CEO-ul Codespring despre reglajul fin", iunie 2013.
3. Diana Ciorba – „HCI (Human-Computer Interaction): On the Verge of Change" - Publicația Codespring „Company Papers", ediția nr. 9, septembrie 2012.
4. John W. Ryan - „Buyer Steps",2011, ISBN-1460988612.
5. P. Kotler, K.L. Keller, M. Brady, M. Goodman, T. Hansen - „Marketing Management"; Education Limited 2009, ISBN – 978-0-273-71856-7.
6. – E. N. Berkowitz, R. A. Kerin, S. W. Hartley, W. Rudelius - „Marketing"; Richard D. Irwin Inc. 1994, Printed in the U.S. A, 1234567890 VH 09876543.

Autor

Diana Ciorba - Marketing Manager @ Codespring

Zornitsa Tomova

Imediat după absolvirea Masteratului de Inginerie și Management al Afacerilor din cadrul Universității Warwick (UK), Zori a co-fondat compania britanico-bulgară 42 IDEAS și, timp de doi ani, a fost consultant pe inovație în servicii și producție, atât pentru companii, cât și pentru startup-uri și ONGuri. Membră a organizației non-profit Start It Smart, a organizat evenimente și programe cu scopul de a sprijini tinerii în începerea afacerilor proprii.

În 2013, ca parte din echipa UseTogether, Zori a format comunitatea de sharing economy din Bulgaria. La începutul anului 2014 a lăsat totul în urmă, pentru a începe ZenQ ('Ze way to say thank you and appreciate your friends. On your mobile. În zeconds.')

12

MANAGEMENTUL PRODUSELOR PENTRU STARTUP-URI: EXPERIENŢA ZENQ

Totul a început într-o cafenea Starbucks din Sofia (Bulgaria), la sfârșitul lui 2013, când hazardul mi-a oferit ocazia întâlnirii cu o persoană care mi-a influențat ulterior evoluția mea profesională. Preocupările profesionale comune au reprezentat obiectul principal al conversației noastre.

I-am explicat că îmi conduc propria companie de consultanţă de mai bine de doi ani, ajutând companiile să dezvolte produse inovatoare în IT, producţie, servicii și alte sectoare. De asemenea, conduceam proiecte care ofereau oamenilor sprijin în începerea afacerilor, ca membru al Start It Smart NGO. De asemenea, ajutam startup-ul UseTogether din Cluj să își construiască comunitatea în Bulgaria. Și una și alta. Cum spunem noi în Bulgaria, „fie ca ultimul să închidă ușa".

La rândul meu, am aflat că venea din Israel și construise câteva afaceri care făcuseră sute de milioane. Acum căuta oameni care să îl ajute să o dezvolte pe următoarea – o aplicaţie pe mobil foarte simplă care să îţi permită să salvezi poze pe telefoanele prietenilor sau ale familiei tale. Nu aveam nicio idee despre *mobile* la acea vreme (nu aveam smartphone, pentru ca să evit dependenţa de reţele sociale, jocuri și în general dependenţa de mobil). De la această simplă conversație, am ajuns la o colaborare, în urma căreia am reușit să fac foarte multe progrese: am învăţat cum să învăţ, cum să fiu imperfectă şi să fiu OK cu asta, cum să visez şi să îmi urmez visele. Tocmai această ultimă lecţie a fost cea care în final m-a făcut să încetez colaborarea şi, cu un mic impuls din partea unui prieten din Cluj, să încep ceva nou care imediat s-a transformat din vis în realitate.

ÎNCEPE ZENQ

Eu şi cu Mircea am început să „punem la cale" ZenQ – o aplicaţie socială pentru mobil pentru oamenii interesaţi de dezvoltarea personală şi gândirea pozitivă, pentru a spune „mulţumesc" şi a-şi aprecia prietenii extraordinari. Partea cea mai interesantă era că profilul îţi va fi construit de către prietenii tăi care îţi dezvăluiau punctele forte şi faptele pozitive, după cum au fost văzute prin ochii lor. Am făcut puţină cercetare, am derulat câteva mici experimente care implicau aplicaţii *funny paper* şi *stickies*, am desenat câteva cadre (*wireframes*). Dar la Startup Weekend Cluj, lucrurile au luat cu adevărat viteză. Ne-am întâlnit cu nişte oameni extraordinari. Ei au utilizat codul şi alte tipuri de magie pentru a aduce la viaţă în două zile, un prototip funcţional. Am câştigat locul al doilea împreună şi numai două luni mai târziu, *beta* noastră publică era lansată pe iOS şi Android. Unii dintre noi am lucrat doar *part-time*, nu ne-am văzut faţă în faţă în tot acest timp (deoarece Mircea şi cu mine eram în TechPeaks People Accelerator în Italia), şi totuşi, ne-am mişcat cu o viteză incredibilă. O jumătate de an mai târziu, încă dezvoltăm şi rezolvăm probleme mai mici sau mai mari împreună. Avem mai mult de 1200 de utilizatori *beta* şi interes din partea unor investitori din UK şi Bulgaria, ceea ce ne-a făcut să avem o scurtă prezentare la Facebook HQ în Londra, recent.

LECŢII ÎNVĂŢATE

Chiar dacă oferisem consultanţă multor startup-uri și companii în legătură cu dezvoltarea produselor, ZenQ a fost prima mea încercare în domeniul mobil și de a conduce o echipă de dezvoltare a produsului. Da, aceasta înseamnă multe greșeli și lecţii învăţate. Şi totuşi, mai înseamnă şi că, dacă eşti pregătit să ţi le asumi, este mai uşor ca niciodată să iei nişte oameni deosebiţi în echipă şi să începi un proiect propriu. Niciodată nu va exista un moment mai potrivit, nu ai nevoie să ştii nimic mai mult decât ştii acum, vei învăţa ceea ce trebuie pe parcurs. Partea cea mai grea este să perseverezi printre eșecurile multiple și să faci ca lucrurile să meargă și după entuziasmul inițial. Durează cel puţin șapte ani să construiești o afacere în IT care să poată fi susţinută global şi nu există o cale mai uşoară. O întrebare bună pe care să ţi-o pui înainte de a începe – Vreau să investesc următorii 10 ani din viaţa

mea pentru a face asta să funcţioneze? Dacă răspunsul este nu, s-ar putea ca acesta să nu fie proiectul tău. Mai caută. Dacă răspunsul e da, nu ai nimic de pierdut dacă începi chiar acum.

Urmează câteva dintre lecţiile pe care le-am învăţat și care sper că vor fi folositoare atât colegilor startup-eri, cât și celor care se gândesc să construiască un startup într-o zi. Și poate, cine știe, managerii de produs care își câștigă din asta existenţa ar putea beneficia de o perspectivă diferită. Învăţăturile se împart în patru categorii, mentalitate, echipă, proces și utilizatori- Cvartetul Sfânt al startup-urilor la început, dacă doriţi. Există industrii mari care se adresează problemelor legate de dezvoltarea personală, sudarea și managementul echipei, procesele afacerii, marketing, fără a mai menţiona startup-urile. Şi totuşi, când vine vorba de Managementul produselor (Product Management) în contextul startup, există câteva lucruri specifice pe care să le avem în vedere, pe care vi le pot împărtăși din experienţa mea personală. Lectură plăcută!

MENTALITATE

Dezvoltarea unui produs extraordinar începe din inimă. Aici avem trei puncte cheie. În primul rând ceva simplu – tu ar trebui să iubeşti ceea ce faci, deoarece numai o persoană fericită are perseverenţa să creeze lucruri uimitoare. Totuşi, ar trebui de asemenea **să iubești și *oportunitatea* care vine odată cu dezvoltarea produsului**. Un produs nu are limită superioară în privinţa valorii pe care o poate aduce – poţi ajunge la sute de milioane de oameni și le poţi face vieţile mai frumoase. Dacă nu visezi la scară mare, dacă nu ești îndrăgostit de această oportunitate imensă, nu are nici un sens să realizezi un produs. Poţi fi mult mai fericit să lucrezi la altceva care este mai aproape de inima ta – să locuiești într-o casă în pădure, să călătorești în jurul lumii, să furnizezi servicii extraordinare oamenilor, orice. În final, dacă nu eşti sigur că aceasta este ceea ce îţi dorești, nu uita că depinde de tine să faci alegerea. **Dragostea și pasiunea se construiesc, nu ţi se oferă gratis.** Nu poţi iubi pe cineva de la prima vedere – dragostea adevărată crește cu timpul și efortul pe care îl investești în ea. La fel este și cu produsele. Stabilește-ţi un *reminder* zilnic care să îţi amintească de ce faci asta și ce visezi să obţii. Și apucă-te de treabă. Dragostea ta va crește în fiecare zi. Și odată cu ea şi succesul tău.

Unde să îţi stabilești obiectivul? Buddha oferă un sfat foarte bun în legătură cu acest lucru. Imaginează-ţi că desenezi o persoană pe o coală de hârtie și apoi câteva cercuri concentrice în jurul său. Persoana aceea ești tu, cercul cel mai apropiat este familia ta, următorul cerc reprezintă prietenii tăi, apoi colegii, orașul tău, ţara ta, lumea, etc. . Buddha spune că prin stabilirea unui obiectiv mai aproape de ultimul cerc din desenul tău, tu generezi mult mai multă energie în tine însuţi, iar această energie te va ajuta să ajungi acolo. Așa că fii curajos, nu te mulţumi cu obiective care sunt mai prejos de visele tale cele mai îndrăzneţe.

Alege-ţi credinţa cu înţelepciune. A apărut o întreagă industrie care ajută startup-urile să înainteze – cărţi, evenimente, maeștrii guru, acceleratoare. Toată lumea îţi oferă perspectiva sa despre cum ar trebui să îţi dezvolţi startup-ul, iar acest lucru este nemaipomenit. Este o bogăţie fără precedent la îndemâna ta, și totuși este și un pericol. De exemplu, mișcarea *lean* este încă pe val în timp ce vorbim. Ţineţi minte, totuşi, că **marile startup-uri au fost construite înainte de cartea „The Lean Startup". Și vor fi construite mult după ce trece acest val**. Am întâlnit oameni care au progresat uimitor urmând această carte. Și alţii care au urât-o din tot sufletul și care și-au construit calea spre succes urmând alte credinţe proprii. Soluţia este să citești din răsputeri și să fii deschis la toate perspectivele, până când faci alegerea ta legată de ceea ce să crezi. Am descoperit că este bine să fii puţin „schizofrenic" în legătură cu asta și să îţi dezvolţi capacitatea de a avea puncte de vedere și credinţe conflictuale în minte (ascultă-ţi clienţii și nu îi asculta, planifică-ţi traiectoria și resursele financiare și nu le planifica, renunţă și perseverează, etc.). Nu știi niciodată care dintre ele va fi cea mai folositoare în următoarea provocare pe care o vei înfrunta pe drumul tău.

Concentrare! Concentrare! Concentrare! Hai să fim serioși, nu poţi construi Microsoft, Apple și Facebook în același timp. Una dintre calităţile cheie ale unui manager de produs este să spună „nu". Dacă nu îţi dezvolţi rapid această capacitate, te vei pierde în fluxul de idei ale utilizatorului, mentorului, echipei și propriile tale idei, înainte de a-ţi da seama. Lista de funcţionalităţi ale produsului tău actual va continua să se extindă până când va exploda și se va transforma într-o gaură neagră, din care nu va mai ieși nimic niciodată. Totul pornește de la tine și de la început – dacă viaţa ta este o harababură de proiecte, vei avea o harababură în fiecare dintre proiecte.

Alege unul și concentrează-ţi toată energia și atenţia pe acesta.

Greșeli ZenQ, vol. 1: Acum e rândul meu să admit o greșeală – poate voi o veţi putea evita. Iniţial, am presupus că restul echipei noastre va dori aceleași lucruri ca și mine, că vom avea mentalităţi asemănătoare. Greșit. Mi-a luat ceva timp să înţeleg că **fiecare dintre noi avea priorităţi diferite, obiective și planuri diferite** și să mă adaptez la situaţia asta. Fiecare dorește să înveţe şi să se dezvolte în propriul său fel şi are propriile sale motive pentru a participa la proiectul tău. Scopul tău nu este să îi faci „să se alăture forţei întunecate" sau așa ceva, ci să îi înţelegi. Diferenţele dintre voi sunt o binecuvântare deghizată – cu cât sunteţi mai diferiţi, cu atât mai bine pentru startup-ul vostru. Cei care nu sunt de acord cu tine, cei care au ca prioritate altceva în acest moment, cei care vor să facă multe lucruri, aduc o grămadă de experienţă și perspective diferite la masa de lucru. Acesta este un beneficiu extraordinar oricând ai nevoie de o metodă creativă de a soluţiona o problemă cu care te confrunţi, ceea ce se întâmplă în cea mai mare parte a timpului.

ECHIPA

Pur și simplu, ai încredere în ei. Oamenii care se implică în startup-uri sunt extraordinari, și când le dai mână liberă, întotdeauna te vor surprinde. Există multe exemple în acest sens, în experienţa mea. Claudiu, care acum este dezvoltatorul nostru *backend*, nu avea idee de Python (tehnologia pe care o utilizăm) și iniţial nu a avut un rol clar în echipă. O lună mai târziu, a văzut că avem o lipsă pe partea de *backend*, a învăţat să scrie cod în Python și a început să execute în mai puţin de două săptămâni. Mihai, care lucrează la iOS, s-a oferit voluntar să prezinte proiectul nostru în TSM și să ne ajute cu interfaţa de înscriere a utilizatorului. Cristi a jucat un rol important iniţial în definirea a ceea ce trebuia făcut în termeni tehnici și mai târziu s-a oferit să facă aplicaţia noastră Android (nu avea experienţă în domeniul *mobile*). De asemenea, ajută mult **să nu te amesteci în cod** – ai încredere în băieţii de pe partea tehnică că vor face ca totul să meargă. Treaba ta este să te asiguri că produsul final funcţionează după cum aţi discutat. Cum se ajunge la acest lucru, este treaba expertului din fiecare domeniu.

Dinamica succesului. Acesta este singurul aspect pe care milionarul pe care l-am menţionat la început îl căuta la o echipă. Formarea ta, experienţele,

potenţialul nu contează. Singurul lucru care contează este să îi dai gata pe ceilalţi împreună, să ai „dinamica succesului". **Echipa ta fie se împacă sau nu se împacă, iar aceasta e vizibil chiar de la început**. Mi-a spus că văzuse oameni extraordinari care au eșuat în lucrul împreună în câteva startup-uri pe care el le-a creat, dar şi anumite combinaţii destul de ciudate de oameni care acţionau magic asupra altora. Cum se spune, „echipa A" este mai bună decât un grup de „jucători A". Nu are sens să analizăm motivele, pur și simplu așa se întâmplă. Când drumul devine dificil (ex. păstrarea utilizatorilor fideli nu este chiar pe roz, precum ţi-ai dori), tu ai nevoie de energia, creierul și creativitatea tuturor pentru a defini pașii următori. Creierul tău nu poate egala niciodată creierul colectiv al 2-5 persoane diferite care vor să facă ceva să funcţioneze.

Funcţionează echipele dispersate? Și încă cum! Cu Skype/Google Hangouts, Trello și Google Drive, orice este posibil. Noi am livrat primul nostru MVP după două luni de utilizare a acestora pentru a comunica între Cluj și Trento (Italia) , și a funcţionat mai bine decât oricare echipă *offline* cu care am lucrat vreodată. Atunci când oamenii vor să facă ceva, o fac și nu găsesc scuze. O echipă dispersată înseamnă că oamenii lucrează când vor ei (fără a aștepta să te aduni laolaltă pentru a face ceva), ei pun întrebări și primesc răspunsuri rapid, fără a avea nevoie de contact direct, totul poate fi căutat și realizat în timp ce vorbești. Ceea ce este important e că managerul de produs este întotdeauna disponibil pentru a discuta deciziile/problemele importante și pentru a se asigura că toată lumea se întâlnește suficient de des pentru ca să poată exista colaborarea.

Greșeli ZenQ, vol. 2: Nevoie de mai multă bere. Îmi doresc să fi numit un Ofiţer Şef pentru Băutură la început. Este inconvenientul echipelor dispersate – există pericolul să te concentrezi pe muncă și este mai greu să îţi organizezi activităţile care nu au legătură cu munca. Inclusiv, în primul rând, băutura, de unde, pe lângă râsete și istorii personale, se nasc multe idei grozave. Aceasta nu poate fi imitată online. Ar fi nemaipomenit dacă o parte din echipă care se află într-un loc s-ar întâlni în mod regulat. În orice caz, ce puteţi face este să încercaţi să vă întâlniţi faţă în faţă măcar o dată pe lună, dacă se poate. Merită cu vârf și îndesat!

PROCESUL

Nu planifica drumul. Există o tendinţă naturală să vrei să ai totul clar dinainte să începi. Totuși, adevărul este că în primele luni ale proiectului tău, înainte de a ajunge la potrivirea produsului cu piaţa, ideea ta se schimbă constant. Este un proces aleatoriu – un coleg de echipă tocmai a avut o idee extraordinară în timpul implementării, te-ai lovit de o limitare tehnică, ai aflat despre un competitor care are o strategie interesantă, un mentor ţi-a dat o idee grozavă despre capitalizare, ai citit „Hooked" și te-a lovit ideea că, pentru a reţine utilizatorii, trebuie să ai notificări, e-mailuri care să le amintească de tine. **Există un flux constant de modificări mici și simple care sfârșesc prin a-ţi transforma produsul în ceva foarte diferit faţă de ce ai avut în minte când ai început.** Deci ce poţi face în această privinţă? Trei lucruri. Să ai un plan de cel mult cinci paşi în minte (ex. să obţii investiţia, să aduni echipa, să te muţi în US, să obţii un parteneriat cu Facebook, să faci minuni). Dezvoltă cu echipa ta o definire detaliată și foarte clară a funcţionalităţilor pentru versiunea la care lucraţi în prezent. Și o idee generală cu privire la principiile/ valorile principale înspre care tindeţi cu produsul vostru, cu o prioritate clară pe care vreţi să o abordaţi în următoarea versiune. Este suficient.

Începe rapid și murdar. Eu sunt un mare fan al formatului Startup Weekend datorită culturii pe care o promovează chiar de la constituirea echipei tale. Te face să te concentrezi pe viteză, execuţie și distracţie chiar din prima zi – toate elementele de bază ale unei culturi a startup-urilor de succes. Pe lângă asta, un weekend de muncă intensivă împreună spre un obiectiv clar arată fiecărui membru din echipa ta ce înseamnă să faci lucruri împreună. Fiecare dintre voi vede dacă există o „dinamică a succesului" în echipa voastră sau nu. Şansele sunt ca să aveţi un mic câştig care va suda echipa şi vă va motiva să continuaţi împreună, într-o altă încercare. Nu trebuie să aşteptaţi următorul Startup Weekend, multe startup-uri au weekenduri de lucru care funcţionează de minune. Ceea ce este important e ca tu, în calitate de manager al produsului, să fii foarte clar cu privire la cele 2-3 funcţionalităţi care reprezintă baza aplicaţiei tale. Schiţează trei ecrane pe o bucată de hârtie și asigură-te că toată lumea înţelege, oferă *feedback* și este de acord cu ceea ce tu hotărăști că poate fi făcut. Cheia este să livrezi ceva mic și simplu, cât mai curând posibil.

Produsul minim viabil (MVP) nu este nici produsul minim atrăgător (MLP) și nici produsul minim cumpărat (MBP). Ok, deci acum echipa ta este dornică să treacă mai departe la următoarele etape și să realizeze ceva mai mare. Din nou, gândiți simplu și rapid și aici – chiar și pentru cea mai simplă definire de produs, vor exista atât de multe lucruri care să meargă prost încât nu va mai părea atât de simplu. Cu cât este mai mare și mai complex, cu atât crește probabilitatea ca echipa ta să obosească luptând cu toate acele probleme mici și s-ar putea să nu mai livrați niciodată. Și**, indiferent ce construiești, indiferent cât de mult timp ți-a luat și cât de mult ai vorbit cu clienții înainte, există o șansă de 99% ca oamenii să nu se îndrăgostească de produsul tău atunci când îl lansezi**. Numai utilizatorii reali care folosesc soluția ta zilnic, pe măsură ce acordezi și testezi lucruri diferite cu ei, numai ei te pot îndruma. Numai după ce ai livrat MVP-ul îți poți da seama care este produsul tău minim atrăgător, care va fi acel set de experiențe și funcționalități care îi face pe utilizatori să arate ca niște înfometați care revin mereu la borcanul tău cu bunătăți. Odată ce ai descoperit asta, va fi în sarcina ta să îți dai seama care este produsul minim cumpărat – setul minim de lucruri pentru care utilizatorii vor plăti, astfel încât să te poți menține în afaceri și să crești. Dacă ești suficient de ambițios în visurile tale și ai succes în realizarea acestei treceri graduale de la MVP la MBP, vei avea o estimare de sute de milioane de dolari a startup-ului tău în acel moment. În etapa MVP, **este foarte important să stabilești** așteptările echipei tale. Lansarea voastră nu va aduce mulțimi de utilizatori care se vor călca pe picioare pentru a încerca produsul. Chiar dacă ați avut un articol în TechCrunch sau ceva similar, veți începe foarte bine, după care utilizatorii vor înceta să mai revină. MVP-ul vostru reprezintă punctul de pornire în adevărata muncă de găsire a utilizatorilor și de a-i face să rămână! Și toată lumea din echipa ta ar trebui să fie pregătită să facă față situației.

Măsoară totul de la început. Dacă nu măsori, nu înveți și nu ai un startup. Simplu. Dat fiind că MVP-ul tău este sursa ta de învățare, nu îți poți permite să nu pui parametri clari pentru ceea ce dorești să înveți, să îi urmărești săptămânal, să efectuezi teste AB, să măsori impactul fiecărei îmbunătățiri și funcționalități noi. Nu este greu să implementezi așa ceva. Totuși îți va lua o zi întreagă să te gândești care vor fi indicatorii de performanță cheie (KPI) și ce alte lucruri vei măsura la fiecare ecran/interacțiune pentru

a-ţi permite să vezi de ce un indicator de performanţă nu este așa cum ţi l-ai dorit tu să fie. Dacă aș putea să mă întorc în timp, m-aș asigura că am implementat nu numai Flurry, dar și MixPanel, Localytics, Google Analytics pentru mobile. Astfel, poţi să vezi ușor tot ce poţi face cu acestea și poţi alege ce e mai potrivit pentru tine. Altfel, sfârșești prin a visa la anumite informaţii (cum ar fi comportamentul utilizatorului individual sau abilitatea de a trimite e-mailuri pe baza acestuia) și a aștepta următoarea lansare pentru a le obţine. Aceasta te încetinește fără motiv.

Paznic la o turmă de iepuri. Câteva cuvinte despre organizare. Agile este baza, dar fiţi atenţi cum implementaţi aceste principii. Scopul este să funcţioneze pentru voi şi nu voi să munciţi pentru aceasta. Luaţi ce aveţi nevoie, experimentaţi și observaţi ce impact are asupra muncii voastre. Pentru noi, trei lucruri au funcţionat precum magia. *Sprint*-urile – un volum specific de muncă care produce un produs testabil pentru o perioadă dată (o săptămână, 2, 4 săptămâni). Trello este extraordinar pentru așa ceva – trebuie doar să descompui MVP-ul tău în funcţionalităţi, să decizi la care dintre ele să lucrezi în primul tău *sprint*, să o divizezi în sarcini pentru toată lumea din echipă (ex. UX, *backend*, iOS, Android) și poţi porni la drum. Al doilea element Agile – *user stories* – așa definești fiecare sarcină în Trello: „**ca utilizator (începător), doresc să (văd X pe profilul meu), astfel încât să pot (face Y)**". Aceasta face minuni cu privire la cât de bine înţelege toată lumea ceea ce construiești și poate rezolva problemele pe parcurs. Și în final – *scrum*-uri regulate: întâlniri scurte în care fiecare răspunde la trei întrebări: „ce am făcut în ultima perioadă?; ce probleme am avut?; ce vreau să fac în continuare?". Acestea vă oferă un ritm de lucru regulat, un loc în care să cereţi ajutor, să discutaţi, să rezolvaţi probleme împreună, să fiţi la curent cu ce fac toţi ceilalţi. E foarte important!

Greșeli ZenQ, vol. 3: Iniţial, am dat greș cu durata *sprint*-urilor. Toată lumea lucra *part-time* și totuși am lucrat cu *sprint*-uri lungi de 1-2 săptămâni, ceea ce îi extenua pe băieţii de la tehnic. Același lucru se aplică și la Scrum – iniţial, ne întâlneam mult prea des (la fiecare două-trei zile), pierzând timpul cu discuţii lungi care urmau fiecărei sesiuni de scrum. Ne-am mulţumit apoi cu *sprint*-uri lunare și *scrum*-uri săptămânale, care par a fi cele mai potrivite în cazul nostru. Aceste durate pot face diferenţa dintre un bici și o tobă – între ceva care te stresează și îţi dăunează vieţii și ceva care doar introduce un ritm în vâslitul echipei tale. Contează.

UTILIZATORII

Întreabă-ţi și nu-ţi întreba utilizatorii. Nu este o coincidenţă că am lăsat utilizatorii pe ultimul loc aici. Valoarea pe care ei o percep din ceea ce faci este singura metodă de a aprecia cât de bine funcţionează împreună mentalitatea ta, echipa și procesul. Dacă faci bine lucrurile, ei sunt o sursă uimitoare de învăţare, inspiraţie și motivaţie pentru a merge mai departe. Ai nevoie de o bază pe care să construiești, iar această bază este ceea ce vrei să faci, mentalitatea ta, echipa ta și procesele. Așezi baza la locul ei și continui să primești *feedback* de la utilizatorii tăi pentru a te asigura că creezi experienţa pe care ei o doresc. În detaliu, nu dacă este o casă sau o navă spaţială. De multe ori, utilizatorii îţi vor spune un lucru, iar atunci când au produsul în mână vor face cu totul altceva. Vei auzi sute de idei în legătură cu ce ar trebui îmbunătăţit. Sute de idei despre cum poate fi aplicat în cazuri diferite de utilizare. Atunci când primești feedback *cantitativ*, acesta va fi bazat fie pe ceea ce există deja, fie nu va fi suficient de profund pentru aţi construi viziunea produsului în jurul său. Când primești *feedback* calitativ, acesta va fi nesemnificativ din punct de vedere statistic, zgomotos și greu de sintetizat. Nu te poţi baza pe utilizatori ca să îţi spună care ar trebui să fie viziunea produsului tău. Te poţi baza pe ei să te corecteze pe parcurs, să îţi ofere idei despre funcţionalităţi *cool* și să te ajute să înveţi. Ascultă-i în legătură cu acestea!

50% dintre oameni nu vor înţelege. Dacă ești norocos. Şi cel mai probabil aceşti oameni nu sunt ţinta ta – depinde de tine să îţi dai seama de ce, acest lucru te va ajuta să te orientezi mai bine înspre ceilalţi %50 rămaşi. Întreaga afacere Red Bull este creată în jurul a %50 dintre utilizatori, cărora le place. Restul oamenilor urăsc chiar şi gustul său, dar asta nu împiedică Red Bull să facă miliarde de la primul grup. Nu îi poţi mulţumi pe toţi, aşa că cel mai bun pariu al tău este să te concentrezi pe punctele luminoase. Aceştia sunt utilizatorii tăi fideli, cei care iubesc ceea ce faci tu şi care vor să te ajute să îmbunătăţeşti produsul. Privind la experienţele lor uzuale cu produsul tău (cum arată fluxul lor exact, pas cu pas), tu poţi observa tiparele și îţi poţi dezvolta interfaţa utilizatorului astfel încât să îi încurajezi și pe alţi utilizatori să facă aceiași paşi. Pentru asta, ai nevoie de analitică la nivel individual (ex. oferită gratuit de MixPanel).

Experimentează din răsputeri. Ajută să începi întotdeauna cu câteva experimente offline (ex. noi am făcut un test al aplicaţiei pe hârtie cu 30 de oameni înainte de a construi ceva). Chiar și mai târziu, este bine să experimentezi înainte de a construi funcţionalităţi noi majore. Aici cheia este să observi comportamentul mai degrabă decât opiniile. Nu explica, urmărește! Și din nou – are sens numai dacă măsori. Imaginează-ţi care ar fi comportamentul ideal pe care ai vrea să îl vezi, împarte-l în bucăţi mai mici, gândește-te la o metrică pentru fiecare etapă, stabilește o normă pentru ea și vezi cum se compară rezultatele reale cu aceasta. Pe măsură ce avansezi cu MVP-ul tău, fă cât de multe teste A/B poţi – pentru copia ta, pentru butoane, culori, flux, etc. . Există câteva instrumente grozave pentru aceasta, chiar pentru *mobile* (ex. noi folosim Vessel). Vei găsi mulţi băieţi care construiesc încă astfel de soluţii, care se află într-o etapă timpurie și sunt dispuși să negocieze preţurile în schimbul *feedback*-ului. Folosiţi-vă de asta.

Nu explora, persistă. Una dintre cele mai mari neînţelegeri legate de mișcarea *lean* este despre termenul „validare". Iar şi iar vedem oameni care fac dezvoltarea clienţilor, „încercând" lucruri pentru a vedea cum funcţionează. Apoi spun că nu au primit validare, drept o scuză pentru a renunţa la un proiect după ce au fost demotivaţi de rezultatele timpurii descurajatoare sau disfuncţiile iniţiale ale echipei. Primele semne de validare reală au loc atunci când abordezi reglarea produs-piaţă. Dacă dezvolţi ceva apropiat de o soluţie existentă cu o ușoară zvâcnire nouă, aceasta se poate întâmpla rapid (dacă ai un plan solid pentru a te diferenţia de competitori). Dacă faci ceva mai inovativ, îţi va lua multe iteraţii (și noroc) să ajungi acolo. **Dacă vei renunţa de la primul hop din drum, ce rost mai are să începi?** Dacă produse ciudate cum sunt Snapcat sau Yo pot ajunge la milioane de utilizatori, „ideile proaste" care nu pot obţine dragostea utilizatorilor pur și simplu nu există. Totul ţine de execuţie. Să faci sau să nu faci. Nu există încercare.

Greșeli ZenQ, vol. 4: Iată una pe care nu am rezolvat-o nici până în ziua de azi – nu am păstrat relaţiile cu utilizatorii suficient de apropiate. Nu vorbesc de postarea unor poze cu pisici drăguţe pe Facebook, ci despre ceea ce se întâmplă DUPĂ ce un utilizator vine la tine cu *feedback* și idei. Este sarcina ta să menţii acea relaţie, să implici persoana respectivă mai mult, să îi arăţi dragostea ta și să obţii ajutorul său pe parcurs. Necesită efort să construiești

acest proces, să definești ce să întrebi/discuţi în fiecare săptămână și să faci conversaţia să continue. Am ajuns la concluzia că acest aspect este de fapt un rol în sine, pe care ar trebui să îl aibă chiar de la început cineva din echipa ta, care să fie clar răspunzător pentru aceasta. Nu toţi cei 500 de oameni care ţi-au scris să îţi spună ce le place sau nu le place în legătură cu *beta* ta atunci când a fost lansată, vor avea timp să discute în continuare. Dar cei 20% care o vor face, sunt mai mult decât suficienţi pentru a te ajuta când testezi lucruri noi și te îndrepţi spre ajustarea produs – piaţă.

CONCLUZII

În acest articol am arătat că, dacă eşti o fată de 25 de ani fără nici un pic de experienţă în dezvoltarea produselor, nimic nu te poate împiedica să schimbi acest lucru. Același lucru este valabil și pentru voi, oriunde aţi fi și oricare este experienţa voastră. De fapt, lumea întreagă și progresul nostru ca și umanitate depinde de voi să riscaţi, să ieșiţi din zona voastră de confort și să faceţi ceva ce alţii nu au mai făcut până acum. Probabil că veţi pierde, dar dacă inspiraţi pe altcineva să vă urmeze și el/ea reușește, milioane de oameni s-ar putea să o ducă mai bine în câţiva ani. Ce ai putea dori mai mult?

Lecţiile împărtășite aici s-au adresat celor care sunt la început de drum și s-au referit la mentalitate, la cum să îţi construiești o echipă, ce procese să urmezi și la cum să comunici cu utilizatorii. Acestea sunt în întregime supuse criticii, discuţiilor și interpretării din partea voastră. Poate că am greșit în totalitate în legătură cu unele dintre ele și voi afla aceasta săptămâna viitoare sau mâine. Vă doresc să îmi descoperiţi greșelile pe măsură ce avansaţi și faceţi propriile voastre greșeli (dacă deja aţi găsit câteva, știţi despre ce vorbesc).

Dacă există ceva ce ar trebui să reţinem, sper să fie o mică mantra pe care am învăţat-o de la tipul din Israel care a făcut minuni pentru mine. Este o replică simplă pentru a-ţi învăţa subconștientul care ar trebui să fie atitudinea ta faţă de toată nebunia care te așteaptă în legătură cu startup-ul.

Alege un moment din zi (poate dimineaţa, poate înainte de evenimente sau întâlniri importante, etc.) în care să te privești în oglindă și să spui: **„Ești o persoană extraordinară. Poţi face orice. Și te iubesc."** Și apoi ieși în lume și fă minuni.

Vă mulţumesc! ZenQ!

Autor

Zornitsa Tomova - Cofondator @ ZenQ

Sergiu Damian

Sergiu Damian este un architect software experimentat, cu o profundă înțelegere a designului *software*, practicilor și proceselor de dezvoltare și, nu în ultimul rând, al dinamicii umane din echipe. După 13 ani petrecuți în diverse roluri în companii locale de *outsourcing* a ales să devină un arhitect *software* independent, ajutând diverși clienți în a-și atinge succesul. Este activ în comunități IT, oferă asistență studenților și îi invață pe alții despre *software architecture.*

8

ARHITECTURĂ PENTRU STARTUP-URI. PROVOCĂRI ȘI DECIZII.

În ce fel arată rolul arhitectului care participă într-un start-up? Așa cum un birou mare de arhitectură va face extrem de rar o căsuță de locuit, atunci când poate să își pună numele pe o sală polivalentă, la fel în firmele mari de software, chiar specializate în *outsourcing*, avem de-a face de obicei cu aplicații de tip *enterprise*. Cerințele se discută în ședințe care implică reprezentanți de la client, atât tehnici cât și din domeniul funcțional, iar discuțiile sunt centrate pe înțelegere cât mai precisă. În procesul dezvoltării aplicațiilor, arhitectul ia decizii sub presiunea unor factori multipli de calitate, a efortului și a altor constrângeri.

Valorific experiența recentă care mi-a arătat un proces similar derulat într-un start-up și vă propun să analizăm împreună cum s-a schimbat demersul și rolul de arhitect în acest context.

După aproape cincisprezece ani petrecuți ca dezvoltator, arhitect sau manager în firme de *outsourcing* clujene, am primit recent o provocare nouă din partea unui prieten. Visul lui e de a realiza o suită de aplicații mobile care să ajute pe utilizatorul potențial să își îmbunătățească calitatea generală a vieții. Rugămintea lui a fost să fiu eu arhitectul, omul care gândește partea tehnică pentru acest ecosistem. Am acceptat fără să clipesc. De ce? M-am întrebat adesea ce e diferit într-un context de start-up comparat cu mediul

enterprise cu care eram obișnuit eu. Echipa sa în acel moment nu avea niciun programator. Era formată doar din oameni care dezvoltau un concept, o idee, un alt fel de a ajuta oamenii. Pentru mine era o noutate și din punctul de vedere al conținutului era vorba de o suită de aplicații departe de zona obișnuită de aplicații de prelucrare a datelor care, într-un fel sau altul, facilitează diverse procese din mediul *enterprise*.

Vă invit în continuare să vedem împreună cum a decurs micul nostru experiment oprindu-mă mai apăsat asupra a trei decizii, care cred că ilustrează cel mai bine diferența între stilul de lucru și arhitectura de start-up față de cea din mediul corporativ.

COMUNICAREA

Prima constatare a fost că în echipă nu exista o imagine clară asupra a ceea ce vrem să realizăm, dar exista dorința acută să avem cât mai repede o primă aplicație. Așa că, deși unii ar spune că asta nu e treaba arhitectului, am reformulat complet modul de lucru. Am stabilit că, dacă nu vorbim toți aceeași limbă, nu avem cum să fim coerenți, să înțelegem despre ce vorbim și unde vrem să ajungem. Deși instinctul ne împingea să pornim cât mai repede construcția tehnică, ne-am dat repede seama că nu știm suficient și că am cheltui efort mult, cu riscuri mari de a da greș.

Pare banal, privind în retrospectivă, însă e extrem de important să găsim mijlocul potrivit de comunicare, de a nota ideile într-un mod ușor de înțeles pentru toți, cât mai vizual și cât mai aproape de realitatea înconjurătoare. Așa că, am folosit unelte de creat schițe ale interfeței cu utilizatorul. Echipa de concept a adoptat rapid acest stil și, mai mult, a ales singură unealta potrivită, trecând de la Balsamiq (folosită inițial) la Indigo Studio. În scurt timp, toată echipa adoptase acest limbaj și ideea ne era tuturor din ce în ce mai clară. Întâlnirile noastre au devenit mai eficiente, structura aplicațiilor devenea clară, ne era ușor să ne imaginăm cum vor funcționa și chiar ce ar putea simți un utilizator.

În ultimii ani, uneltele specializate pentru crearea și editarea (unele permit chiar rularea) schițelor de aplicații (în engleză *mockups* sau *wireframes*) au evoluat în calitate, flexibilitate și ușurință în utilizare. Aș îndrăzni să spun că oricui cu sau fără studii de IT, i-ar fi acum la fel de ușor să creeze schițe de aplicații ca și cum ar edita un document oarecare. Așadar, orice proiect

software poate începe acum cu o schiță.

Cred cu tărie că arhitectul ca liant între funcțional și tehnic poate și chiar are datoria să ajute întreaga echipă și nu doar partea ei tehnică. Până la urmă, și arhitectura echipei e tot o arhitectură, nu? Am mai întâlnit și în lumea proiectelor de mari dimensiuni încercări de a desena aplicația înainte de a începe implementarea ei, însă, de multe ori, proiectul e atât de mare încât efortul pare gargantuesc sau chiar zadarnic. Chiar și atunci, aș zice că a schița esența aplicației sau părțile ei mai interesante ajută în comunicare, iar comunicarea face ca mecanismele proiectului să se potrivească mai bine.

CALITATEA

A doua provocare de care m-am lovit a fost să-mi redefinesc așteptările tehnice. Eram obișnuit să analizez în detaliu cerințele clientului (adesea o organizație reprezentată de câțiva indivizi) legate de performanță, scalabilitate, securitate, adaptabilitate și alte astfel de atribute (răspund la întrebarea Cum?), în plus față de cele strict funcționale (răspund la întrebarea Ce?). Mi-am dat seama că viitorul utilizator al aplicațiilor create de noi va avea cu totul alte criterii de judecată, cele mai importante fiind, foarte probabil, valoarea adusă lui și ușurința în utilizare (efortul depus pentru a obține acea valoare).

Împreună cu echipa, am decis că în contextul nostru e mai important să aducem aplicația în fața utilizatorilor decât să răspundem acum tuturor întrebărilor posibile despre calitate. Am ales, astfel, în cunoștință de cauză și asumându-ne investiții ulterioare, să implementăm aplicația folosind cea mai simplă soluție tehnică, dar extensibilă ulterior. Nu voi intra acum în detaliile tehnice, pentru că același raționament s-ar putea aplica oricărei combinații de platformă și limbaj de programare.

Ne lovim frecvent, mai ales dacă suntem obișnuiți cu proiecte sau produse software de dimensiuni mari, de dificultatea definirii calității. A găsi echilibrul perfect între calitatea și puritatea tehnică, viteza și costul dezvoltării și momentul ieșirii pe piață e un aspect dificil de gestionat. În mediul *enterprise*, calitatea are impact major în costul total al deținerii și operării unui sistem (TCO). Fără să am niște statistici precise, costul calității e adesea mai mare decât costul direct al producerii funcțiilor produsului. Ca *outsourcer*-i, ne-am dori calitate maximă la costuri minime, deziderat care e adesea doar un tru-

ism imposibil de atins.

Dar în context de start-up avem libertatea să alegem. Clientul nu e o firmă mare ci sunt (în cazul nostru) miliardele de posesori de dispozitive *mobile* de pe glob, deci nu ne e imediat vizibil și accesibil. Nu îl cunoaștem, ci doar ne imaginăm că își dorește același produs pe care îl visăm noi. Așadar, singura modalitate de a interacționa cu clientul și de a afla dorințele sale este de a-i pune la dispoziție aplicația și a colecta *feedback*-ul său. Dacă așa stau lucrurile, atunci e mai degrabă important să lansăm produsul, chiar dacă e incomplet sau la un nivel de calitate mai scăzut, decât să încercăm să atingem perfecțiunea, să respectăm toate regulile dar să riscăm să aflăm prea târziu că produsul nostru nu satisface nevoi reale sau nu atinge așteptările utilizatorilor.

Sunt faimoase cazurile Gmail, Flickr și alte câteva care au fost lansate și apoi păstrate în formă „neterminată" (în programare cunoscută sub numele de „beta") ani de zile. Motivul este unul legat de gradul de satisfacție al echipei de dezvoltare legat de starea produsului, nicidecum de calitatea lui sau abilitatea de a-și deservi utilizatorii.

În cazul nostru, am decis să renunțăm deocamdată la unele aspecte ce țin de calitatea serviciilor de pe server (calitate pur tehnică) ca să putem investi mai mult în calitatea experienței utilizatorului. Citind povestea lui Jeff Bezos și a Amazon (apărută în limba română la editura Publica) întâlnim relatări picante ale reversului medaliei, și anume efortul necesar de a suplini aceste compromisuri atunci când succesul te „lovește" în mod neașteptat.

PLANUL

Odată stabilit modul de comunicare în echipă și de documentare a intențiilor noastre, puteam să demarăm construcția. Ne-am dat seama după primele estimări că efortul necesar, raportat la timpul disponibil era disproporționat. Această disproporție se năștea din faptul că fiecare dintre membrii echipei folosește pentru a participa o felie limitată de „timp personal". Am decis, cum menționam mai devreme, să renunțăm la unele aspecte tehnice. Cu toate astea, complexitatea nu poate fi eliminată complet. A face *software* nu e lucru simplu. Astfel, am întâlnit o a treia provocare. Aceea de a planifica munca întregii echipe (incluzând aici concept, design grafic și dezvoltare) în așa fel încât să reducem timpul nefolosit și să maximizăm viteza cu care

mergem cu toții înainte.

De data asta mi-a fost mai ușor. Reveneam într-o zonă familiară. Mi se părea chiar mai simplu decât într-un proiect mare, pentru că vorbeam de o echipă mai mică, de funcționalități mai puține și dependințe mai clare. Am stabilit așadar împreună cu echipa, că vom adopta strategia plugurilor de zăpadă care bat pârtiile sau curăță iarna pista unui aeroport înzăpezit. Vom decala ciclurile în așa fel încât echipa de concept să fie cu un pas înaintea celei de design, respectiv echipa de servicii (*back-end*) să fie cu un pas înaintea celei de aplicații. Știu, veți zice că e la mintea cocoșului, însă a atinge ritmul corect și a gestiona patru fire diferite nu e chiar ușor pentru cel ce joacă rolul de manager de proiect. În plus, nu e chiar evident ce înseamnă a fi cu un pas înaintea celuilalt. Noi am decis că finalizăm toate cele necesare celeilalte echipe pentru a putea lucra fără întrerupere de „marți" încolo (orice zi a săptămânii e la fel de potrivită pentru acest scop). Important e ca toți să înțelegem și să dansăm același stil de vals.

Una din responsabilitățile arhitectului *software* este de a defini calea spre atingerea succesului dorit. Adesea, în proiectele *enterprise*, nu se poate estima sau planifica fără ca arhitectul să fie cel puțin consultat. Și asta pentru că soluția tehnică aleasă (pentru care arhitectul este responsabil) influențează major costurile, planul, etapele și durata realizării produsului.

Dar sunt multe alte variabile, dincolo de soluția tehnică, care influențează realizarea unui proiect *software*: experiența echipei, condițiile de lucru, uneltele de care dispun, implicarea clientului, gradul de motivare, uneori chiar și vremea de afară. Cum unele dintre ele nu sunt controlabile sau chiar predictibile, influența lor crește complexitatea proiectului.

Complexitatea inerentă produselor *software* e atât de mare încât Uncle Bob (Robert C. Martin , un faimos consultant *software* și autor de cărți) a postulat în una din cărțile sale că e imposibil de estimat și planificat corect munca mai departe de două săptămâni. Totuși, e posibil să înțelegem etapele mari, dependințele și ordinea în care să construim, având o estimare aproximativă suficient de bună pentru a putea gestiona așteptările clienților în baza ei.

E dificil să fii flexibil, deschis și curios și să abordezi situația liber de constrângeri, atunci când miza proiectului, măsurată financiar este mare. De obicei, în firmele mari se lucrează cu miză mare și toți ne dorim câteodată să ne „vedem numele pe o sală polivalentă". Ceea ce am aflat sau mi-am reconfirmat, e că, indiferent de rolul avut în echipă, merită câteodată să dezvolți

și o „căsuță de locuit”, măcar pentru bucuria de a reveni la resurse minime și libertate maximă.

Notă: Suita de aplicații despre care vorbim este încă în dezvoltare, ne apropiem cu pași mari de prima lansare, de prima aplicație în versiune *beta*. Deocamdată ne sincronizăm și încă nu a cerut nimeni din echipă să schimbăm formula. Dacă vrei să ne urmărești, o poți face aici:

http://moodtracker.giminiapps.com/

Autor

Sergiu Damian - Independent Software Architect

Silvia Răusanu

Silvia este o tânără pasionată de IT, interesată de *big data* și de inteligență artificială.

Silvia este combativă și este aproape un hazard să o contrazici.Așadar ,dacă pornești o polemică, trebuie să fii înarmat cu argumente puternice bazate pe fapte.

A început să lucreze în ISDC acum patru ani ca *junior developer.* De atunci a avut un parcurs interesant, acum fiind unul dintre experții în *big data,* modelare de date, procesare de date și învățare automată din ISDC. Pe lângă pasiunea ei pentru acest domeniu, Silvia ține prezentări la conferințe, scrie articole de specialitate și oferă *training*-uri.

Silvia a terminat Facultatea de Matematică și Informatică la Universitatea Babeș-Bolyai, Cluj-Napoca, în 2010, iar în 2012 a absolvit masterul Sisteme Inteligente la aceeași universitate.

9

SUB PRESIUNEA *BIG DATA*

SURSE DE PRESIUNE

A fost odată ca niciodată o companie cu o afacere interesantă; compania avea un grup devotat de oameni de afaceri care urmăreau dezvoltarea produsului lor pentru a îndeplini necesitățile clienților lor, dar și pentru a-și menține nivelul veniturilor lor la cel mai bun nivel posibil. De asemenea, compania avea și un departament tehnic, plin de ingineri și analiști talentați care făceau ca visele și planurile afacerii să devină realitate, și care, în același timp, erau mulțumiți și suficient de incitați de munca lor. Acest basm a durat o vreme îndelungată: datele, instrumentele, procesele erau toate acolo pentru a menține lucrurile și pentru a împinge afacerea pe calea succesului. Totuși, nimic nu durează pentru totdeauna. Nici lumea afacerilor și nici cea a IT-ului nu stau pe loc. Odată cu creșterea exponențială a Internetului și a accesului pe care oamenii îl au la tot felul de resurse *online*, un fel de domino a fost pus în mișcare, iar compania a început să simtă presiunea.

Datele din jurul afacerii erau așteptate să crească liniar în volum, an după an, conform modului în care au fost inițial proiectate . Baza de clienți era în continuă expansiune, după cum la fel erau și pretențiile lor. Furnizorii de date au suferit o creștere asemănătoare, fiind capabili să livreze cantități mult mai mari de informații. Procedurile dedicate procesării datelor au fost construite pentru a fi scalabile doar pentru un anumit volum, dar acum acestea nu mai par potrivite și, cu siguranță, nu mai sunt suficient de performante. Presiune!

În jurul lumii, un *graf* social imens a început să se formeze bazat pe platforme precum Facebook, Twitter, Yelp, Youtube și preaputernicul Google. Utilizatorii își exprimă cu mai multă libertate opiniile, oferă *feedback* imediat pentru produsele pe care le folosesc, își modelează personalitățile lor *online* prin exprimarea antipatiilor și a preferințelor. O abordare având în centru produsul, cu o linie de suport pentru clienți este de domeniul trecutului; lumea afacerilor se reorientează către o perspectivă orientată spre client. Clientul își dorește ca lucrurile să fie personalizate doar pentru el și să nu piardă timp cu informații inutile. Compania nu a prevăzut așa ceva, iar acum trebuie să găsească o soluție care să încorporeze toate aceste forme de date în afacerea lor. Presiune!

Tehnologia avansează într-un ritm rapid și își face apariția în locuri neașteptate: camere video pentru control, senzori de greutate, senzori pentru trafic, etc.. Clienții ar aprecia cu siguranță informațiile care le-ar putea ușura viața cât timp sunt *offline*; totodată, compania ar beneficia de cunoașterea comportamentului clienților lor pentru a lua anumite decizii. Totuși, produsul afacerii nu este pregătit pentru date ce vin cu viteză atât de mare. Presiune!

Oamenii tehnici din companie erau satisfăcuți cu munca lor și erau la curent cu tendințele tehnice. Dar ei aud tot mai des despre "big data", și nu mai sunt atât de fericiți pentru că nu au ocazia să lucreze cu cele mai noi instrumente despre care vorbește toată lumea: NoSQL, Hadoop, etc.. Presiune!

Dacă abstractizezi povestea companiei de basm, chiar poți vedea presiunea generată de volumul, varietatea, viteza și multitudinea de instrumente generate de *big data*. Cu această presiune se confruntă oamenii de *business* și cei tehnici. Citind printre rânduri, devine evident că numărul de lucruri ce trebuie luate în considerare când îndrepți un proiect în zona *big data* nu este neglijabil, iar "presiunea" este împărțită între cei ce iau decizii atât pe partea de business, dar și pe cea tehnică. Nu există rețete care să rezolve tot, dar sunt principii de considerat și întrebări de pus înainte de a începe călătoria de eliberare a presiunii *big data*. Câteva dintre aceste rețete vor fi prezentate în secțiunile următoare.

ELIBERAREA DE SUB PRESIUNE - PERSPECTIVA BUSINESS

Ca om de afaceri, trebuie să înțelegi ce îți pot oferi datele din jurul aplicației tale precum și unde este loc de îmbunătățire a experienței clientului, deci și a veniturilor tale. Apoi să te pui pe treabă.

Cu toate acestea, înainte de a te îndrepta în direcția *big data*, trebuie să te asiguri că celelalte date - *small data* - sunt în ordine. În orice afacere, stagnarea veniturilor sau a bazei de clienți, etc., sau, mai rău, declinul acestora, se poate întâmpla, deși nimic notabil nu s-a schimbat astfel încât să afecteze domeniul afacerii sau clienții. Ai putea să crezi că folosirea *big data* ar putea să-i îndepărteze de această cale descendentă, dar ar fi mai bine ca mai întâi să verifici integritatea și stabilitatea sistemului înainte de a trage vreo concluzie. Dacă serviciul pentru clienți tot primește plângeri pe același subiect și, chiar dacă angajații pot rezolva problema foarte repede, poate ar fi cazul să aloci timp pentru a elimina rădăcina problemei, în loc să investești în mod repetat efortul minim pentru a corecta efectele. Dacă în cadrul proceselor este un pas în care consistența trebuie rezolvată prin acțiuni manuale de alterare - schimbare de status în baza de date, rularea recurentă a unor proceduri pentru a corecta datele - atunci ar trebui să investești în automatizarea acestor procedee, în loc să fie făcute la cerere. Acest tip de comportament în relația *business*-IT denotă că starea *small data* nu este bună, iar adăugarea mai multor date (mai puțin structurate decât cele existente) va aduce doar mai multe pagube și mai puțină fiabilitate. Având aceste premise îndeplinite pentru *small data*, poți face *big data.*

Ca persoană de *business* este foarte important să ai imaginea de ansamblu a direcției în care te îndrepți, dar nu trebuie să te aștepți că implementarea întregii tale viziuni să fie la fel de rapidă ca și crearea unui plan. Chiar mai mult, nu te aștepta ca utilizatorilor să le placă schimbarea la fel de mult cum îți place ție. Dacă ar fi să propun o rețetă ca oamenii de afaceri să profite de trendul *big data*, următorii pași ar sumariza-o.

1. **Analizează direcția de *business* și identifică problema pe care vrei să o rezolvi folosind *big data*.** Nu sări către un trend tehnic - *big data* - fără să ai un caz concret pentru aplicația ta. Încearcă să cuantifici (ipotetic) succesul ideii tale - va îmbunătăți satisfacția clienților tăi? va îmbunătăți retenția de clienți? îți va crește veniturile? este un procent

care merita efortul?

2. **Din problema identificată, taie o felie verticală pentru rezolvare.** Partea aleasă va fi reprezentativă pentru noul produs; trebuie să fie completă în termeni de funcționalitate și performanță dar nu întregul plan proiectat anterior. Unii denumesc aceasta "produs minim viabil" - (eu îi zic "produsul minim iubibil") – pentru că depunând efort minim, trebuie să creezi o parte din produsul complet pe care atât compania ta, cât și clienții tăi, o vor iubi.
3. **Implementează și testează partea aleasă pe un set de date demn de *big data*.** Implementarea e partea cheie, dar testarea e la fel de importantă deoarece atributele calității pentru performață au relevanță mai mare când discutăm de *big data* (cu accentul pe *big*). Deci, chiar dacă soluția este performantă pe un set de date de test, este inutilă dacă pe date de producție nu funcționează. De asemenea, ca parte din faza de testare, este chiar "testul" în producție, confruntarea cu utilizatorii, primirea *feedback*-ului, adevărata evaluare a valorii noului product. Rezultatele din sistemul *live* ar trebui să influențeze deciziile din următorii pași.
4. **Mergi la pasul 1.** Da, este o rețetă cu buclă, pentru că schimbarea este continuă la un produs, dacă vrei să rămâi pe piață. Mai mult, prin *big data*, formatele de date, sursele, frecvența sunt în continuă schimbare și trebuie menținut pasul; și în același timp trebuie mereu încorporat *feedback*-ul de la clienți.

O idee de avut în minte pentru oamenii de afaceri, este ca atunci când iau în considerare *big data*, mai întâi să se asigure că produsul curent sau baza este în regulă și că noua dezvoltare va fi stabilă . O altă idee, conform scenariului de mai sus, este să se miște cu pași mărunți, iar la fiecare incrementare să integreze opiniile adevăraților utilizatori.

ELIBERAREA DE SUB PRESIUNE - PERSPECTIVA TEHNICĂ

Trendul *big data* s-a format mai mult în jurul provocării tehnice pe care o expune - este pur si simplu o realitate pentru unele companii, decât o ofertă irezistibilă pentru afacere; deci "presiunea" e mai mare asupra părții tehnice:

modul de stocare a datelor, algoritmii, instrumentele, arhitectura aplicației trebuie să se schimbe pentru a îmbrățișa și folosi *big data.* Fiecare dintre aspectele enumerate anterior sunt subiectul a numeroase articole, așadar ideile următoare sunt cel mult un punct de pornire.

Cu riscul de a părea o repetiție, trebuie precizat că înainte de a începe un proiect *big data* peste o aplicație existentă, "clasică", trebuie să ai partea de *small data* pusă la punct. Dacă datele ajung în mod constant într-o stare inconsistentă și este necesară intervenție umană în baza de date pentru a le fixa sau dacă în mod constant te izbești de un zid când vine vorba de *flag*-uri, statusuri, etc. sau dacă tu și echipa ta bâjbâiți prin întuneric căutând semnificația anumitor date, atunci nu mai visa la *big data* și ai grijă de datele existente. *Big data*, dacă nu este încadrată într-o bază stabilă, produce dezastre. Ca persoană tehnică, nu trebuie să te gândești că noua tehnologie ce vine cu *big data* va înlocui vechiul sistem de stocare de date, iar datele tale dezordonate se vor potrivi perfect cu noua abordare. O aplicație ce folosește *big data* nu șterge tot ce există, de fapt este construită să completeze funcționalitatea și arhitectura curentă.

Provocarea *big data* este să faci gestiunea corectă pe întregul parcurs al datelor prin aplicație: modelare, stocare, procesare, utilizare. În jurul tuturor acestor pași din ciclul de viață al datelor, foarte multe instrumente au fost create pentru managementul lor, ceea ce face să fie o decizie grea alegerea stivei tehnologice pentru aplicație; de aceea, *tool*-urile precizate nu trebuie luate ca model de bună practică.

MODELARE PENTRU *BIG DATA*

Unele persoane încă presupun că suportul pentru date nestructurate, fără schemă, într-o varietate mare de formate, căruia i se face publicitate constant pentru a fi implementat de instrumentele pentru *big data*, este echivalent cu concedierea tuturor analiștilor de date. Dimpotrivă, modelarea și analiza datelor a devenit mai importantă decât era pentru că o varietate mare de date trebuie să fie validate de oameni: nu are sens să îți aduci date despre păsări dacă aplicația ta e despre acțiuni la bursă. Mai mult, alimentarea unui *tool* cu un morman imens de date, fără nici o indicație de folosire, ar putea rezulta în cel mai bun caz, cu instrumentul potrivit, în obținerea abilității de a căuta orbește prin toate datele. Ca orice alt instrument computațional, *tool*-uril pentru *big data* au nevoie de informații pentru a transforma datele în for-

mat-mașină pentru cele mai bune rezultate.

Modelarea de date în contextul *big data* are nevoie de o altă abordare. Dacă pentru problema clasică, normalizarea era cheia pentru o soluție de succes, fără a lua în considerare funcționalitatea cerută pentru aplicație, în *big data*, cerințele funcționale preiau conducerea prin utilizarea denormalizării "personalizate" pentru aplicație. În timp ce normalizarea sparge datele brute de dragul protocolului, denormalizarea se face doar pentru a servi aplicația. Mai mult, procesul de modelare a datelor ce presupune normalizarea este greu maleabil: odată ce modelul a fost stabilit, în cazul unor probleme de performanță, doar modelul de interogare este ajustat pentru îmbunătățirea rezultatelor. Într-un scenariu asemănător, dar folosind o bază de date denormalizată, procesul este mai agil: atât interogările, cât și modelul suferă alterări pentru a obține cele mai bune rezultate.

STOCARE PENTRU *BIG DATA*

O aplicație modernă, în ceea ce privește stocarea de date, nu se mai bazează doar pe un sigur sistem de stocare, cum ar fi cel mai faimos: bazele de date relaționale. De asemenea, când adaugi în schemă *big data* nu înseamnă că noul sistem ales pentru depozitarea datelor le va înlocui pe toate celelalte. Informații sensibile despre utilizatori (parole, adresa de acasă, informații legate de cardul de credit), detalii despre comenzi (preț, status, informații livrare), etc., nu ar trebui să fie stocate într-un sistem *big data* deoarece acest tip de date este menit să rămână sub incidența și managementul bazelor de date relaționale. Cu alte cuvinte, configurarea pentru stocarea de date trebuie să fie de tip poliglot. Date sensibile și vitale pentru procesul aplicației ar trebui menținute în bazele de date relaționale, datele ce vin în perechi (valuta și cursul valutar, țară și drapel) ar trebui ținute în baze de date cheie-valoare (cum ar fi Redis), în timp ce datele nestructurate ar trebui stocate în baze de date NoSQL, alese în funcție de tipul datelor. Dacă datele tale sunt "similare" cu documentele, atunci ar trebui să alegi o bază de date pentru documente (cum ar fi Cassandra, MarkLogic, MongoDB, etc.); dacă datele tale conțin triple, atunci ar trebui să folosești o bază de date *graf* (Neo4j, Titan, AllegroGraph, etc.).

PROCESAREA *BIG DATA*

Pentru o perioadă destul de lungă de timp, s-a remarcat o confuzie generală legată de echivalența între *big data* și pasul de procesare pentru *big*

data : task-urile *map-reduce* implementate în Hadoop erau considerate a fi cel mai bun mod de a trata *big data*; însă Hadoop este doar un instrument din stiva tehnologică ce poate fi folosită pentru aplicații *big data*. Totuși, vasta răspândire a instrumentelor de executare de *map-reduce* a fost o mare reușită în lumea *big data*; din cauza volumului în creștere de date, procesarea a fost împinsă către o abordare distribuită pentru mai multă eficiență - deci s-ar putea înțelege sursa confuziei. De-a lungul timpului, Hadoop a devenit liderul în ceea ce ține de stocare și procesare *offline* pentru *big data*; chiar și așa, ideea de *offline* începe să se învechească în timp ce alte abordări pentru *map-reduce* prind viteză - Spark, Summingbird sunt exemple de *tool*-uri pentru execuție în timp real de *task*-uri *map-reduce* în memorie.

Pasul de procesare nu se aplică doar o singură dată în ciclul de viață a datelor; procesarea se poate face la momentul ingestiei, declanșat de un eveniment sau bazat pe un program. Scopul procesării este de a curăța datele, de a extrage informații relevante, îmbogățirea datelor cu mai multe informații.

În relație cu modelarea, procesarea poate fi un instrument util de folosit: să presupunem că datele vin dintr-o sursă externă (nu din *ETL*-ul sistemului), iar pentru aplicație este nevoie de o altă structură, derivată din cea originală; în această situație, procesarea la momentul ingestiei, este recomandabilă pentru a avea doar date curate în baza de date. Denormalizarea - oportună pentru acest context - poate cauza probleme de integritate, care trebuie rezolvate ulterior (în cazul în care *master*-ul datelor este în sistemul *big data*). Așadar un *task* de procesare poate fi distribuit pentru a restabili consistența.

Procesarea merge mână în mână cu funcționalitatea aplicației (la fel ca modelul), deci, la început, pasul de procesare era folosit doar pentru operații simple, cum ar fi rapoarte statistice peste seturi prea mari de date pentru a fi acoperite de instrumentele tradiționale. Cu timpul, procesarea a ajuns să fie folosită pentru a extrage cunoștințe dintr-o masă mare de date. De exemplu, entitățile numite pot fi extrase pentru a face parte din triple, iar în acest mod se poate forma un *graf* de relații peste datele existente; sau entitățile recunoscute pot fi marcate în documentul original pentru a oferi mai multă structură și pentru a ușura accesul la informații.

UTILIZAREA *BIG DATA*

O parte a problemei cu *big data* este faptul că volumul presupus este atât de mare încât o nouă stivă tehnologică trebuie aleasă doar pentru a ține în viață procesele mai vechi; însă aceasta este doar tratarea unui simptom nu un

mod de a valorifica ceea ce oferă *big data.* Cu un volum mai mare de date, chiar și statisticile devin mai semnificative și exacte, iar o corelație ce pare o coincidență poate fi demonstrată cu valori - ceea ce aduce mai multe informații pentru *business.*

O altă dimensiune pentru *big data* este varietatea și aceasta ar putea reprezenta cheia pentru a descoperi informații valoroase. Diferite surse de date pot oferi date despre aceeași entitate, deci modele se pot descoperi și în date neuniforme. Asemenea informații pot îmbunătăți reținerea de clienți printr-o experiență complet personalizată per utilizator, nu doar pentru publicitate și marketing, ci și pentru modelul afacerii; să luam de exemplu polițele de asigurare pentru mașini, și modul lor de calculare doar în funcție de niște parametri ficși (cum ar fi vârsta). Cum ar fi dacă comportamentul șoferilor în trafic ar putea fi accesibil prin senzori sau o aplicație? Atunci, poate șoferii tineri ar putea avea polițe mai ieftine pentru că ar putea dovedi cu date că sunt șoferi siguri, în ciuda vârstei lor, iar businessul ar atrage mai mult clientela tânără.

În ceea ce privește algoritmi folosiți, nu sunt prea multe noutăți pentru *big data*; algoritmii de învățare automată primesc în sfârșit atenția cuvenită. Odată cu creșterea cantităților de date din aplicații și cu progresul făcut pe sursele adnotate (pentru procesarea limbajului natural - corpusuri adnotate, tezaure extensive, ontologii), analiza de sentimente, previziunile, clasificarea automată a clienților, ies din laboratoarele de cercetare și pot fi folosite pentru a îmbunătăți afacerea. Într-adevăr, în zona *tool*-urilor o multitudine de opțiuni au apărut (Mahout, OpenNLP, PredictionIO, Jubatus, Encog, etc.) pentru a încuraja uzul cât mai general. În plus, pentru rezultate mai bune și integrarea cu executoarele de *map-reduce*, algoritmii au fost ușor modificați pentru a funcționa și într-o abordare distribuită.

De-a lungul descrierii pașilor pentru managementul *big data*, multe baze de date NoSQL, executoare distribuite de *map-reduce* și *framework*-uri dependente de domeniu au fost precizate, multe dintr ele *open source*, ceea ce nu ar trebui să impună limite în ceea ce privește costul. Însă e chiar așa? Pentru mine cea mai bună caracterizare pentru software *open source* este "libertatea exprimării nu înseamnă bere gratis", dar nu datorită modelelor de licențiere (cum se folosește de obicei). Mie îmi place să duc analogia către relația dintre timp și cost pentru construirea unui produs: ți-ai putea exersa

libertatea exprimării și eventual vei convinge pe cineva să-ți cumpere bere, dar nu ar fi fost mai rapid să o cumperi? Decizia ce trebuie luată în materie de tehnologii se referă la : dacă ai găsit gratis *tool*-urile potrivite pentru a implementa funcționalitatea, dar echipa de dezvoltare trebuie să lucreze la integrare,dar în același timp, există o alternativă cu licență plătită ce necesită mai puțin efort pentru dezvoltare, atunci ce alegi? Argumentul costului nu mai este valabil. Partea mai puțin bună în a avea o multitudine de *tool*-uri la dispoziție este că majoritatea lor nu au timp să se maturizeze, să formeze comunități în jurul lor, pentru că altele apar și pretind a fi mai bune; de aici vine lipsa de standard și de rețete despre cum se poate aborda tehnic *big data.*

PRESIUNE BUSINESS VS. IT

Departamentul de business și cel de IT nu au mereu o relație prea fericită. Oamenii de afaceri sunt un fel de visători la venituri, iar oamenii de IT sunt interesați doar de codul lor și de tehnologiile pe care le pot experimenta. Unde se potrivește *big data* între aceste două părți? Unele persoane cred că datele intră în responsabilitatea departamentului IT și în acest caz, schimbarea *big data* trebuie inițiată de această parte. Adevărul este la mijloc, ambele părți trebuie să se informeze reciproc și continuu despre orice oportunitate de a împinge lucrurile înainte, deoarece responsabilitatea este a amândurora. *Big data* trebuie să fie recunoscută de oamenii mai puțin tehnici, cel puțin ca și conținut și trebuie privită ca orice altă tehnologie: avansează afacerea. De-a lungul prezentării punctelor cheie pentru *big data* pentru partea tehnică, s-a subliniat faptul că în timpul implementării și a luării deciziilor tehnice, cerințele funcționale trebuie să conducă. Așadar - făcând un pas în spate pentru a avea imaginea de ansamblu - ideea este că *business*-ul trebuie să aibă "ce"-ul, iar IT-ul "cum"-ul.

Un punct dureros în relația *business*-IT este că *big data* nu poate salva o companie. După cum s-a explicat mai devreme despre problemele cu *small data*, pot exista și alte probleme de esență cu rădăcini vechi, iar unele persoane pot decide naiv că *big data* va îndepărta toate problemele. Fără a fi un sfat foarte bun pentru aplicațiile *big data*, trebuie reținut faptul că nici o tehnologie nu poate să schimbe modul de gândire al unei companii și să împiedice părăsirea pieței competitive.

ÎN AFARA PRESIUNII *BIG DATA*

Luând în considerare ultimele secțiuni, s-au oferit informații de bază pentru a avansa în lumea *big data* atât pentru business cât și pentru IT. Dar este suficient pentru un plan de lungă durată și o supraviețuire fără stres? Știre bombă: presiunea nu dispare niciodată! În zilele noastre, nevoile tehnologice dar și cele de afacere, avansează într-un ritm alert, deci noi surse de dificultăți pot apărea. Indiferent de mediu, este important să lăsăm partea funcțională să conducă, iar tehnologia îi va urma în concordanță cu nevoile specifice. În același timp, să fim atenți la balanța dintre costul pentru *big data* și profit (fie cel curent sau cel pronosticat).

Toți angajații companiei de poveste au citit acest material plin de informații importante și oamenii de business au început să lucreze mai aproape de IT, iar nivelul de stres al companiei scădea în timp real. Și așa compania a început să lucreze cu *big data* și au operat fericiți până la adânci bătrâneți…

Autori

Silvia Răusanu - Senior Developer @ ISDC
Dan Danciu - Software Architect @ ISDC

Bogdan Rus

Dr. ing. Andrei Bogdan Rus este șef lucrări (lector) în cadrul Universității Tehnice din Cluj-Napoca, Facultatea de Electronică, Telecomunicații și Tehnologia Informației. Devenind din anul 2008 membru al colectivului UC Labs (Unified Communications Laboratories) din cadrul aceleiași facultăți, domnul Rus a participat în proiecte de cercetare internaționale ce aveau ca temă principală tehnici de comunicații în Internetul viitorului. Rezultatul întregii activități de cercetare a fost validat în diverse articole publicate la conferințe internaționale și reviste de specialitate. Adițional activităților de cercetare, Andrei Bogdan Rus este implicat și în activități didactice cu studenții înscriși la cursurile de licență și masterat.

10

TENDINȚE DE EVOLUȚIE A COMUNICAȚIILOR SPRE INTERNETUL VIITORULUI

În ziua de azi, lumea comunicațiilor prin Internet cunoaște o efervescență tot mai crescută, cu un număr din ce în ce mai variat de servicii ce vin în întâmpinarea nevoilor utilizatorilor. În spatele acestor servicii recent apărute se află de cele mai multe ori tehnologii noi ce oferă posibilitatea schimbului de date la viteze din ce în ce mai mari, noi paradigme de accesare a informației și nu în ultimul rând noi metode de utilizare/partajare a resurselor din infrastructurile de comunicații existente.

În cadrul acestui context de cristalizare de noi direcții și tehnologii, apare conceptul de separare a planului de control al rutării de cel al comutației fluxurilor de date. Dacă în abordarea clasică ambele sunt implementate la nivelul echipamentelor de rutare, există în prezent o tendință clară de transfer a mecanismelor de control către un nivel superior, cât mai aproape de aplicații. Această abordare este cunoscută în literatura de specialitate sub numele de SDN (Software-Defined Networking – rețele definite prin *software*). Planul de control din paradigma SDN are acces direct la planul de comutație a datelor, ce înglobează echipamentele din infrastructura de rețea (ex. *router*-e, *switch*-uri și alte echipamente intermediare). Întregul efort de schimbarea a paradigmei rețelelor cu comutație de pachete are ca țel final facilitarea implementării de soluții inovative care să aducă îmbunătățiri majore comunicațiilor prin Internet, în cel mai scurt timp. Printre cel mai des întâlnite aplicații care au adoptat această abordare, putem aminti: controlul dinamic al accesului utilizatorilor în rețea, distribuirea eficientă a încărcării serverelor, virtualizarea resurselor rețelei, eficientizarea consumului de energie a rețelei,

migrarea automată a mașinilor virtuale fără întreruperea serviciilor [1].

Primele implementari comerciale de succes ce au adus tehnologia SDN în lumina reflectoarelor au fost: sistemul de management al traficului în WAN (Wide Area Network), dezvoltat de Google [2] și platforma de virtualizare a resurselor din rețea dezvoltată de Nicira, companie achiziționată de către VMware. În prezent, majoritatea marilor jucători din domeniul tehnologiei informației precum furnizori de servicii de tip *cloud*, furnizori de Internet, producători de echipamente și companii de servicii financiare au format diverse consorții în cadrul cărora își concentrează eforturile în vederea dezvoltării tehnologiei SDN : Open Networking Foundation [4], Open Daylight [5]). Astfel, putem afirma faptul că în prezent se fac pași concreți către realizarea unui vis demult enunțat și anume crearea rețelelor de date cu adevărat programabile care pot să se adapteze fără probleme la dinamica serviciilor disponibile și accesate în ziua de azi prin Internet. Din punct de vedere istoric se evidențiază trei faze ale evoluției spre ceea ce înseamnă în ziua de azi rețele programabile:

1. Din a doua jumătate a anilor `90 și până la începutul anilor 2000 a apărut conceptul de rețea activă ce a introdus funcții programabile în arhitectura rețelelor de calculatoare;
2. În perioada 2001 – 2007 s-a vehiculat ideea separării planului de comutație a datelor de cel de control datorită apariției de interfețe specializate ce au facilitat comunicația dintre cele două;
3. Începând cu anul 2007 apare tehnologia OpenFlow împreună cu diverse versiuni de sisteme de operare deschise, create special pentru echipamente de rețea.

Una dintre aplicațiile tehnologiei SDN considerate a fi cele mai importante la ora actuală și pentru dezvoltarea căreia se alocă în prezent cele mai multe resurse, este platforma de virtualizare a resurselor rețelei. Această fază este următorul pas logic ce este necesar a fi efectuat după apariția soluțiilor de virtualizare a puterii de calcul și a capacității de stocare. Un alt factor important ce accelerează dezvoltarea acestei direcții este apariția pe piață a operatorilor virtuali de telecomunicații. Aceștia utilizează parțial resursele infrastructurilor de comunicații a unuia sau mai multor furnizori, pentru a oferi servicii noi utilizatorilor.

Restul articolului este organizat după cum urmează: capitolul doi demonstrează că tehnologia OpenFlow este cea mai semnificativă din grupul de soluții SDN existente la ora actuală, urmând ca în capitolul trei să prezentăm

o aplicație în care aceasta este folosită cu succes pentru a optimiza *rutarea* datelor în interiorul rețelei. Capitolul patru conține rezultatele experimentale care confirmă avantajele protocolui de *rutare* propus. Capitolul cinci încheie această lucrare cu un set de concluzii ce sintetizează principalele idei ce trebuie reținute vizavi de conceptele prezentate.

TEHNOLOGIA OPENFLOW

Mecanismele de tip SDN fac parte dintr-o arhitectură în care componenta de control a rețelei este decuplată de cea de comutație a datelor și adusă mai aproape de stratul aplicație prin biblioteci de funcții specializate (API – Application Programming Interface). În consecință, se permite optimizarea deciziilor luate în rețea astfel încât să se țină cont de cerințele serviciilor a căror date sunt transmise prin infrastructura de comunicații (vezi *Figura 1*). Un alt avantaj al acestei abordări este faptul că echipamentele din rețea pot fi mult simplificate deoarece acestea nu mai trebuie să implementeze o gamă foarte largă de protocoale deoarece toate deciziile sunt luate de către entitatea de control a rețelei. Singura cerință pentru aceste echipamente este ca ele să accepte și să execute instrucțiuni provenite de la entitatea de control. Astfel, configurarea securizarea și optimizarea infrastructurii se poate face automat cu programe specializate în executarea acestor funcții.

Prima soluție apărută ce permite interfațarea planului de comutație a datelor cu cel de control, în vederea implementării paradigmei SDN, este tehnologia OpenFlow. Aceasta permite accesul direct și controlul echipamentelor de comutație existente în infrastructura de rețea: comutatoare și *router*-e [6].

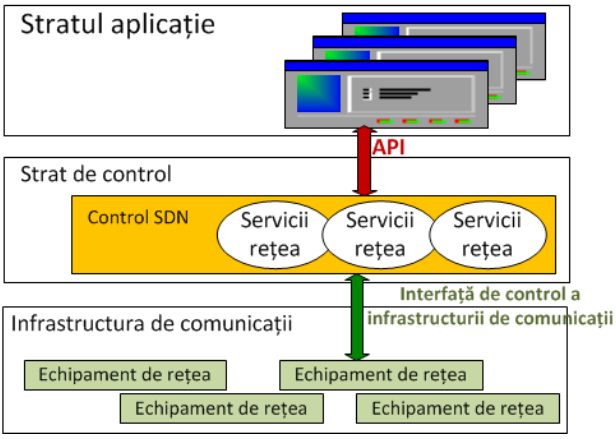


Figura 1. Arhitectura SDN

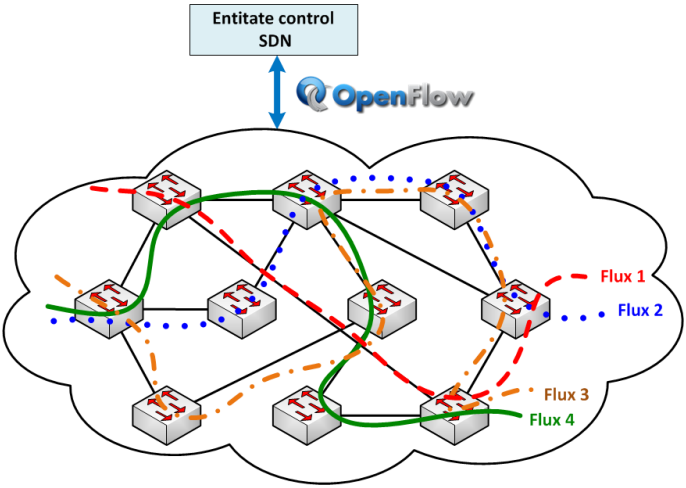


Figura 2. Control la nivel de flux cu ajutorul OpenFlow

În ceea ce privește granularitatea de reprezentare a datelor, se folosește noțiunea de flux ce identifică *unic* date care respectă un set predefinit de reguli. Spre deosebire de *rutarea* clasică, programarea funcționării rețelei la nivel de flux de date oferă un control fin ce permite infrastructurii să răspundă în timp real la toate schimbările ce apar datorită dinamicității aplicațiilor (vezi *Figura 2*). Cheia acestei abordări este tabela de fluxuri ce conține informații referitoare la setul de date ce străbat fiecare echipament în parte [7]. Pe lângă partea de identificare, o astfel de tabelă conține și acțiunea ce trebuie luată atunci când un pachet recepționat este detectat ca făcând parte dintr-un anumit flux de date. Această acțiune specifică de cele mai multe ori interfața de ieșire pe care trebuie redirectat mai departe pachetul în calea sa spre destinație (vezi *Figura 3*).

Utilizat inițial doar în infrastructuri bazate pe tehnologii din familia Ethernet, OpenFlow poate fi în ziua de azi implementat atât în rețele fizice ce utilizează o gama largă de tehnologii cât și în rețele virtuale. Deoarece echipamentele ce suportă rutarea bazată pe mecanisme OpenFlow implementează și protocoale clasice de rutare, adoptarea acestei tehnologii SDN de către furnizorii de servicii se poate face progresiv, menținând costurile la nivele acceptabile.

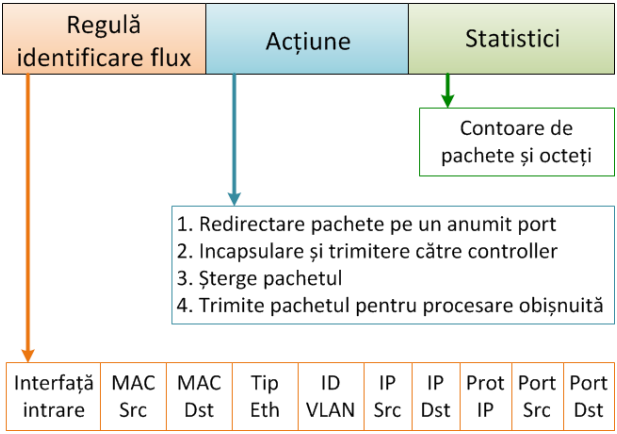


Figura 3. Tabela de fluxuri, existentă în echipamente compatibile OpenFlow

Odată cu apariția tehnologiei OpenFlow au fost elaborate o serie de soluții de control a rețelei ce se bazează pe aceasta: NOX, Beacon, Helios, Big Network Controller, SNAC și Maestro. Primul mecanism apărut ce se încadrează în această categorie a fost **NOX**. Acesta este de fapt un sistem de operare cu rol de control a rețelelor ce conțin echipamente de comutație OpenFlow, oferind suport pentru aplicații scrise în limbajele C++ și Python. O a doua soluție disponibilă pentru control este Beacon. Aceasta are ca principală caracteristică posibilitatea de a permite extinderea funcționalității sale prin adăugarea de module scrise în limbajul Java. De asemenea, are la bază *framework*-ul OSGi (Open Service Gateway initiative) ce permite aplicațiilor de control să fie pornite, oprite sau reinițializate fără a pierde conexiunea cu infrastructura controlată. Destinat îndeosebi grupurilor de cercetare, Helios este o soluție de gestiune a funționalității rețelei elaborată de către NEC în limbajul de programare C. Aceasta oferă un interpretor de comenzi (en. *shell*) programabil destinat executării de experimente automatizate cu un grad ridicat de complexitate. Big Network Controller este un utilitar proprietar bazat pe Beacon și elaborat de către compania BigSwitch. Acesta oferă o interfață în linie de comandă destinată controlului rețelei. Bazată pe NOX, soluția SNAC face parte din categoria programelor publicate sub licență GNU (GNU's Not Unix!). Acesta oferă o interfață intuitivă și ușor de utilizat pentru controlul punctelor de comutație din rețelele de calculatoare folosind limbajul FML (Formal Modelling Language) cu ajutorul căruia se pot specifica politici de funcționare. Utilitarul Maestro elaborat de

către Rice University ca entitate de orchestrare a funcționării rețelei, este bazat pe limbajul Java și oferă posibilitatea de extensie a acestuia prin module adiționale adăugate în funcție de necesități. O altă caracteristică este faptul că a fost optimizat pentru rularea în paralel a mai multor fire de execuție în vederea creșterii performanțelor de control.

Printre beneficiile ce pot fi dobândite în urma utilizării tehnologiei OpenFlow am putea enumera [6]:

1. Creșterea securității rețelelor prin definirea la nivel înalt a politicilor corespunzătoare. Prin intermediul tehnologiei OpenFlow, acestea sunt translatate în instrucțiuni specifice fiecărui echipament din rețeaua controlată. În cazul în care apar schimbări ale infrastructurii operaționale, actualizările se vor executa într-un timp mult mai scurt decât în abordarea clasică. Deoarece tehnica SDN oferă o vizibilitate globală asupra rețelei, se asigură faptul că setările de control acces, inginerie a traficului și calitate a serviciilor sunt implementate unitar în întreaga infrastructură, inclusiv campusuri, centre de date, filiale distante.
2. Controlul centralizat al unei infrastructuri de rețea multi-vendor în vederea punerii în funcțiune și a configurării cât mai rapide a acesteia.
3. Controlul bazat pe flux permite o gestiune granulară a întregii rețele astfel încât se pot specifica politici la nivel de utilizator, echipament sau aplicație. Acest fapt permite furnizorilor de servicii *cloud* să ofere suport pentru izolarea traficului, securitate și management elastic al resurselor în scenarii în care mai mulți utilizatori partajează aceeași infrastructură.
4. Creșterea satisfacției utilizatorilor. Acest beneficiu este datorat faptului că parametrii ce reflectă în timp real proprietățile rețelei sunt puși la dispoziția aplicațiilor de nivel înalt, astfel încât deciziile acestora să fie optimizate la situația actuală. Un exemplu concludent în acest sens ar putea fi eficientizarea serviciilor de tip IPTV (Internet Protocol Television) care transmit fluxuri video în timp real. Cunoscând starea rețelei și rata de transfer disponibilă, acestea pot modifica dinamic rata la care se face compresia fluxului astfel încât să adapteze automat cantitatea de date transmise pe canal.
5. Accelerarea inovației prin implementarea unor noi modele de business. De exemplu, oferirea de servicii de tipul *IT-as-a Service* în care este necesară posibilitatea modificării dinamice a modului de

funcționare a infrastructurii astfel încât să fie îndeplinite nevoile aplicațiilor și ale utilizatorilor.

6. Reducerea complexității de operare prin automatizarea proceselor de management a rețelei. Astfel, se pot minimiza costurile operaționale și se pot elimina perioadele de instabilitate apărute datorită erorilor inserate de către factorul uman.

Mecanismele de tip SDN și în consecință tehnologia OpenFlow apare ca răspuns la tendințele de virtualizare a resurselor de calcul, dorința de îmbunătățire a mecanismelor de mobilitate și apariția soluțiilor de tip *IT-as-a-Service* (infrastructură IT ce este utilizată doar la nevoie), care exercită o presiune semnificativă, căreia rețeaua clasică nu îi poate face față. Astfel, este nevoie ca aceasta să se poată adapta în timp real la cerințele dinamice provenite din partea utilizatorilor, devenind astfel o platformă optimizată pentru livrarea eficientă a unei game variate de servicii.

OPTIMIZAREA RUTARII UTILIZÂND COMUTAȚIA DE ALGORITMI

Nu putem vorbi de o rețea dinamică ce are capacitatea de a se adapta la necesitățiile utilizatorilor și a serviciilor transportate, fără să abordăm diverse aspecte de eficientizare a *rutării*. În consecință, în acest capitol vom prezenta o soluție ce își propune să ofere posibilitatea de implementare a mecanismelor dinamice de *rutare*.

Paradigma după care a evoluat lumea rețelelor de calculatoare de-a lungul timpului presupunea elaborarea a câte unui protocol distinct pentru rezolvarea fiecărei probleme de comunicații. Se poate înțelege astfel motivul pentru care în ziua de azi exista o gamă largă de protocoale ce trebuie activate pe echipamentele din infrastructura rețelei în vederea implementării unui anumit set de politici. În ceea ce privește *rutarea*, există protocoale de *rutare* dedicate fiecărui scenariu ce poate fi întâlnit: protocoale de tip vector distanță, protocoale bazate pe starea legăturii, protocoale intradomeniu și protocoale ce se ocupă cu *rutarea* datelor între domenii autonome diferite. Fiecare dintre acestea folosec un anumit algoritm pentru luarea deciziilor de rutare (vezi *Figura 4*). Înțelegerea și utilizarea tutoror acestor soluții a devenit o sarcina dificilă pentru administratorii rețelelor, crescând astfel costurile operaționale

pentru astfel de infrastructuri.

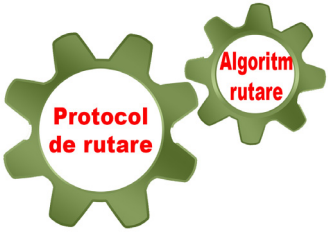


Figura 4. Relația dintre protocoalele de rutare și algoritmi în abordarea clasică

În întâmpinarea acestei probleme, propunem unificarea tuturor protocoalelor de rutare într-unul singur ce poate fi utilizat în toate scenariile posibile. Acesta este capabil să își adapteze comportamentul în funcție de cerințele fiecărui flux de date în parte. Cu alte cuvinte, deciziile de *rutare* nu mai sunt luate de către un singur algoritm, indiferent de necesitățiile curente. GRAS (Gearbox-like Routing Algorithms Selection) [8] este soluția propusă a fi răspunsul la problema enunțată anterior și presupune punerea la dispoziția protocolului de *rutare* a unui set de mai mulți algoritmi, din care va fi activat la un moment dat cel mai potrivit pentru scenariul în cauză (vezi *Figura 5*).

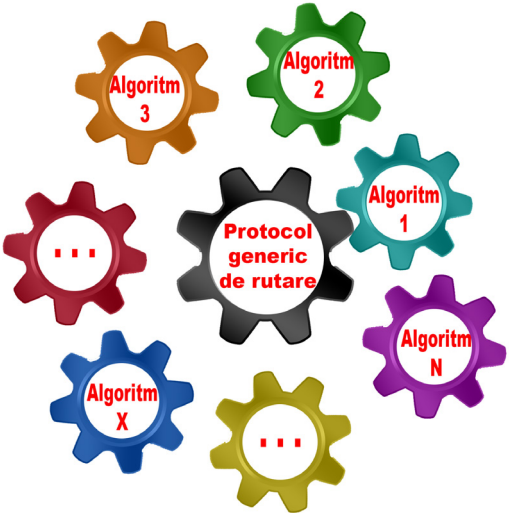


Figura 5. Ilustrarea conceptului GRAS

Algoritmii prevăzuți a fi utilizați într-un astfel de sistem adaptiv sunt următorii (setul poate fi extins în funcție de necesități cu efort minim, deoarece modificările ce trebuie aduse vor fi implementate doar pe elementul de control centralizat a rețelei cu capabilități OpenFlow):

a. Floyd-Warshall [9]: utilizat pentru găsirea căii optime între toate perechile de noduri sursă-destinație dintr-o topologie oarecare. Acest algoritm este recomandat a fi utilizat de către operatorii de infrastructură de telecomunicații pentru a oferi tuturor abonaților cea mai bună cale spre destinație.
b. Dijkstra modificat [10]: este optimizat pentru găsirea căii optime dintre un nod sursă și mai multe noduri destinație într-o infrastructură dată. Acest model de comunicație este recomandat în scenariile în care se dorește găsirea căilor optime de la un furnizor de servicii (transmisii video la cerere, televiziune prin Internet, furnizor de servicii *cloud* etc.) către clienții acestuia.
c. Ford-Fulkerson [11]: acest algoritm poate fi folosit cu succes în cazul în care ruta optimă dintre un nod sursă și un nod destinație nu are capacitatea de a suporta rata de transfer a datelor ce trebuie transmise între cele două noduri. În consecință, este necesară utilizarea acestui algoritm de *rutare* pe căi multiple pentru a găsi toate traseele disponibile între sursă și destinație.

În funcție de starea rețelei și a cerințelor de performanță provenite din partea utilizatorilor și a serviciilor transportate, se va alege algoritmul optim din lista de mai sus. Deoarece infrastructura de comunicații are proprietăți dinamice în timp, este nevoie ca măsurători specializate să se efectueze în timp real astfel încât deciziile luate să fie în concordanță cu starea actuală.

REZULTATE EXPERIMENTALE

Pentru a ilustra fezabilitatea ideilor enunțate în capitolul anterior s-a utilizat o topologie de test formată din patru echipamente pe care rulează soluția Open vSwitch 1.3 (acesta este un *switch* virtual ce implementează tehnologia OpenFlow). Managementul în timp real al rețelei s-a făcut de pe un calculator adițional pe care rulează soluția Beacon (prezentat în capitolul 2) și utilizând o infrastructură de rețea dedicată schimbului de date de control. Traficul util trimis prin rețeaua de test a fost transmis între o stație cu

rol de sursă și una cu rol de destinație (vezi *Figura 6*).

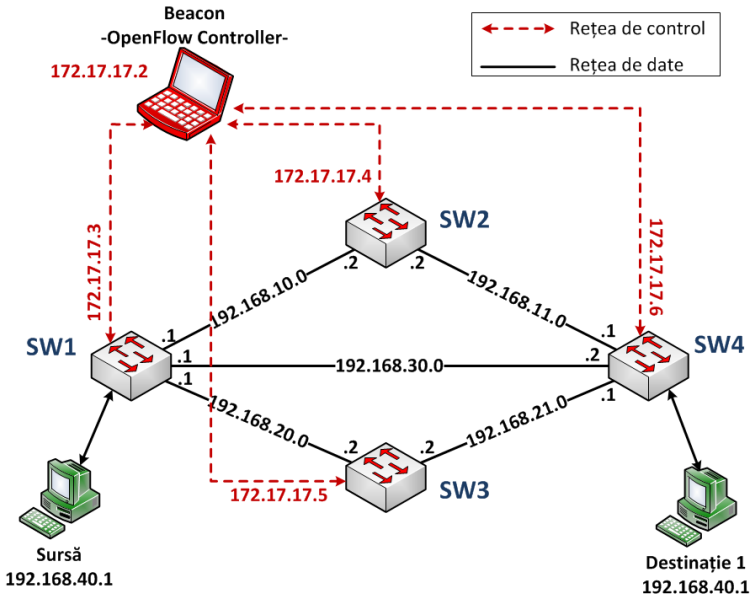


Figura 6. Topologia de test utilizată

Nodurile SW1, SW2, SW3 și SW4 sunt configurate să efectueze comutația astfel: de fiecare dată când un pachet este recepționat, se încearcă determinarea fluxului din care face parte. Dacă o acțiune a fost definită vizavi de fluxul în cauză, aceasta va fi executată (pachetul va fi transmis pe portul de ieșire corespunzător, determinat cu ajutorul algoritmului de rutare activat pentru situația respectivă), în caz contrar, pachetul este încapsulat și trimis mai departe către entitatea de control (Beacon). După procesarea acestei informații se va trimite către nodul de comutație acțiunea ce trebuie aplicată pentru următoarele pachete primite, ce fac parte din acest nou flux.

Scenariul de test implementat presupune transmisia între nodurile Sursă și Destinație a șase fluxuri (F1, F2, .. , F6) a câte 30 Mbps fiecare, către nodul Destinație. Acestea au fost pornite la un interval de 20 secunde unul de celălalt. Marcăm momentele de timp astfel: T1, T2=T1+20s, …, T6=T5+20s. În acest scenariu s-a decis utilizarea algoritmului Dijkstra modificat pentru identificarea celei mai bune căi între sursă și destinație. Deoarece numărul de fluxuri crește în fiecare pas, se observă faptul că din pasul 3 elementul de control decide folosirea algoritmului Ford-Fulkerson pentru utilizarea în pa-

ralel a mai multor căi de transmisie a datelor. Rezultatul cu privire la alocarea fiecărui flux pe rutele existente, poate fi vizualizat in *Tabelul 1*.

Ruta	T1	T2	T3	T4	T5	T6
1-2-4	-	-	F3	F3	F3, F5	F3, F5
1-3-4	-	-	-	F4	F4	F4, F6
1-4	F1	F1, F2	F1, F2	F1, F2	F1, F2	F1, F2

Tabel 1. Alocarea fluxurilor de date pe fiecare rută

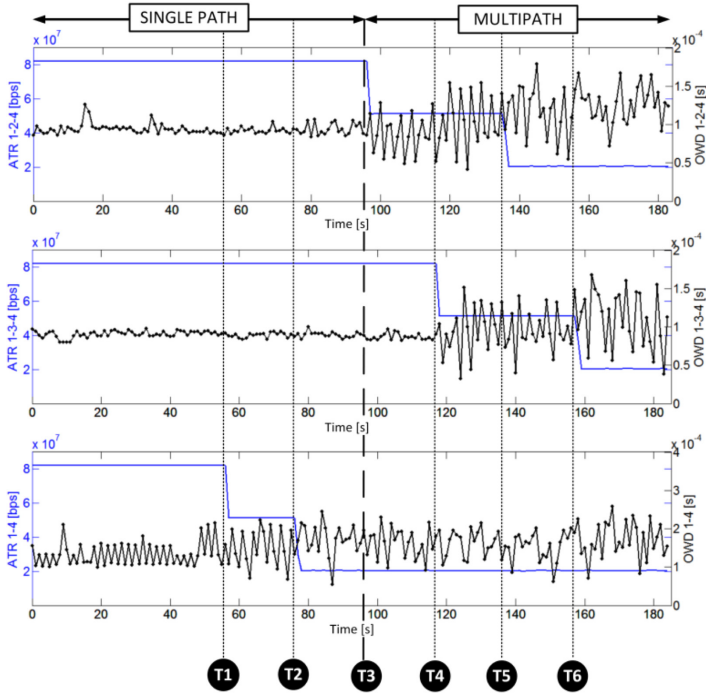


Figura 7. Testul 1: ATR și OWD

În *Figura 7* sunt ilustrate graficele ATR (Available Transfer Rate – rata de transfer disponibila pe o anumită rută) și OWD (One-Way Delay – latența măsurată pe o rută) pentru cele trei căi existente în topologia de test. Se observă faptul că de la un moment la altul starea rețelei se modifică datorită

apariției de noi fluxuri de date trimise către destinație. Utilizând algoritmul Ford-Fulkerson se evită astfel utilizarea excesivă a unei singure legături din rețea (fenomen întâlnit la protocoalele de rutare clasice). Traficul este alocat dinamic pe toate căile disponibile din infrastructură, reușindu-se astfel întârzierea momentului de apariție a fenomenului de congestie.

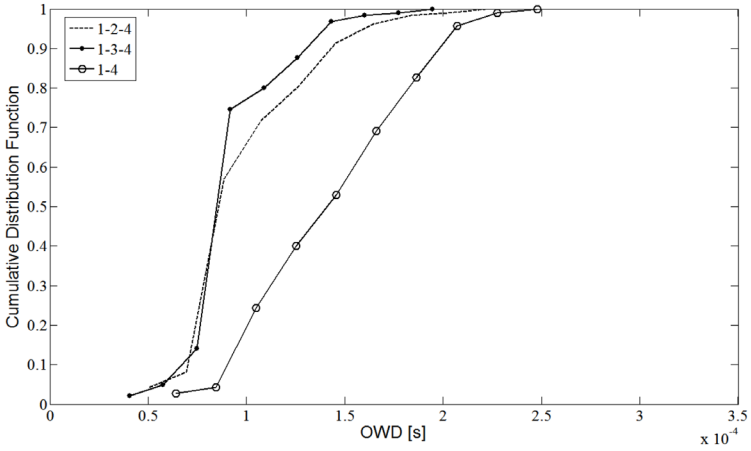


Figura 8. Funcția CDF pentru valorile OWD pe cele trei rute

Rezultatele ilustrate în *Figura 8* demonstrează faptul că implementându-se un comportament adaptiv al rețelei la situația actuală se pot menține parametri de calitate ai serviciilor (latența) la un nivel mulțumitor chiar și în cazul în care rețeaua începe să fie utilizată până aproape de capacitatea maximă. În consecință, acest aspect se va reflecta prin păstrarea unui grad ridicat de satisfacție a utilizatorilor infrastructurii de comunicații.

CONCLUZII

Evoluția către o lume a rețelelor programabile, capabile să se adapteze în timp real la schimbările ce pot apărea, nu se poate face fără separarea planului de control de cel al comutației. O soluție viabilă în acest sens este OpenFlow, care reușește să abstractizeze funcționarea rețelei la un nivel la care o aplicație poate prelua controlul acesteia. În consecință, comportamentul rețelei va fi optimizat la cerințele existente indiferent de dinamicitatea acestora.

Scenariile ce pot beneficia de pe urma capabilității unei infrastructuri de a fi programabilă, sunt nenumărate. Printre cele de importanță ridicată se numără aplicațiile de optimizare a *rutării* în funcție de necesitățile reale ale utilizatorilor și ale serviciilor utilizate de către aceștia. În acest articol am demonstrat faptul că utilizând un protocol de rutare ce folosește adaptiv, în funcție de situația dată, un anumit algoritm de calcul a căilor optime, putem eficientiza modul de utilizare a resurselor, păstrând astfel un grad ridicat de satisfacție. Astfel de abordări vor putea fi testate și mai apoi implementate în rețelele de producție din viitor într-un timp relativ scurt, de îndată ce tehnologiile SDN vor fi adoptate la scară largă.

REFERINȚE BIBLIOGRAFICE

1. [1] N. Feamster, J. Rexford, E. Zegura, „The Road to SDN: An Intellectual History of Programmable Networks", ACM SIGCOMM Computer Communications Review, Aprilie 2014.
2. [2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh,
3. S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hlzle, S. Stuart, and A. Vahdat. „B4: Experience with a globally deployed software defined WAN", *ACM SIGCOMM*, Aug. 2013.
4. [3] ***, „It's time to virtualize the network", Nicira Whitepaper 2012, http://nicira.com/en/network-virtualization-platform
5. [4] ***, Open Networking Foundation, www.opennetworking.org
6. [5] ***, Open Daylight, http://www.opendaylight.org/
7. [6] ***, "Software-Defined Networking: The New Norm for Networks", Open Networking Foundation ONF White Paper, April 13 2012
8. [7] N. McKeown, et al., "OpenFlow: enabling innovation in campus networks", *ACM SIGCOMM Computer Communication Review*, 38 (2), April 2008, 69-74
9. [8] A.B. Rus and V. Dobrota, "Case Study of a Gearbox-Like Routing Algorithm Selection in Runtime", *18th IEEE LANMAN 2011*, Chapel Hill, North Carolina, pp. 1-6, DOI: 10.1109/LANMAN.2011.6076938
10. [9] A.G. Furculita, M.V. Ulinic, A.B. Rus, and V. Dobrota, "Implementation Is-

sues for Modified Dijkstra's and Floyd- Warshall Algorithms in OpenFlow", *12th RoEduNet International Conference*, Constanta, Romania, 2013, pp.50-55, DOI:10.1109/RoEduNet.2013.6714208

11. [10] A.B. Rus, V. Dobrota, A. Vedinas, G. Boanea, and M. Barabas, "Modified Dijkstra's algorithm with cross-layer QoS", *ACTA TECHNICA NAPOCENSIS, Electronics and Telecommunications*, vol. 51, no.3, 2010, pp. 75-80.
12. [11] P. Sevastian, A.B. Rus & V. Dobrota, "Simulation of the Ford-Fulkerson Algorithm Using OMNET++", *ACTA TECHNICA NAPOCENSIS, Electronics and Telecommunications*, ISSN 1221-6542, Vol.53, No.2, 2012, pp. 23-29

Autori

Dr. ing. Bogdan Rus - Șef lucrări @ Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei, Universitatea Tehnică din Cluj-Napoca

Dr. ing Virgil Dobrota - Profesor @ Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei, Universitatea Tehnică din Cluj-Napoca

Zsolt Alfréd Polgár

Zsolt Alfréd Polgár a obținut diploma de Inginer cu specializarea Telecomunicații, Master în Tehnici Avansate în Telecomunicații și Doctor în Inginerie Electronică și Telecomunicații de la Universitatea Tehnică din Cluj-Napoca în 1995, 1996, respectiv în 2002. Din 1996 este angajat la Departamentul de Comunicații al Universitatii Tehnice din Cluj-Napoca, unde în prezent este conferențiar. Domeniile de cercetare de interes implică tehnici de codare a canalului, tehnici de codare a rețelelor de date, rețele radio, tehnici de comunicații prin cooperare, arhitecturi avansate de rețele de comunicații. El a făcut parte din grupurile de cercetare ale programelor COST 289 și COST 2100, a fost membru al echipelor de cercetare ale proiectelor FP7 4WARD și FP7 CODIV și a fost coordonator local al proiectului FP7 UCONNECT.

11

PLATFORMA DE CONECTIVITATE PERMANENTĂ PENTRU TRANSPORTUL PUBLIC INTELIGENT

Asigurarea conectivităţii permanente pentru vehiculele de transport în comun și pentru pasagerii acestora este un obiectiv important în contextul dezvoltării sistemelor de transport inteligente, a tehnologiilor de comunicaţii ale Viitorului Internet și a orașelor inteligente. O soluţie de conectivitate eficientă și ușor implementabilă se bazează pe o arhitectură construită peste reţele radio multi-tehnologie și multi-operator. Soluţia dezvoltată și prezentată aici exploatează resursele de transmisie oferite de reţelele radio eterogene fără a utiliza informațiile de stare disponibile în reţelele de transport ale operatorilor. Proiectarea arhitecturii se realizează pe trei nivele distincte şi anume: sistem, funcţional şi platformă. De asemenea, sunt stabilite principiile de proiectare ale mecanismelor necesare pentru funcţionarea sistemului de conectivitate.

IDEEA DE BAZĂ

Conceptul de Oraș Inteligent primește atenţie din ce în ce mai mare atât din partea comunității de cercetare, a administrației publice cât şi a industriei. Unul dintre aspectele esențiale pentru realizarea unui oraș inteligent este transportul public, care permite oamenilor şi lucrurilor să se miște și care este unul dintre factorii cheie care influențează eficiența globală a unui oraș. Sectorul de transport poate fi îmbunătățit semnificativ prin utilizarea tehnologiilor informației și comunicațiilor, în prezent având loc mai multe inițiative în acest sens. Aceste îmbunătăţiri privesc atât serviciile de transport în sine cât și serviciile oferite pasagerilor, iar o condiție necesară pentru a pune în aplicare aceste servicii este oferirea conectivității permanente în mobilitate [1].

Utilizarea rețelelor publice mobile (3G/4G) pentru asigurarea serviciilor de comunicaţii pentru aplicaţiile specifice Orașului Inteligent nu este totdeauna soluţia cea mai bună. Capacitățile de transmisie ale acestor tehnologii pot fi insuficiente pentru volumul de informații care este în creștere continuă și care trebuie să fie transmis în anumite condiții de fiabilitate și disponibilitate impuse. Acest lucru este valabil mai ales în situația de față, când conexiunea la Internet trebuie să fie furnizată pentru pasagerii unui autobuz (30 - 40 utilizatori) și pentru alte terminale care necesită rate de bit relativ mari (ex. camere de supraveghere, afișaje de informare, etc.).

O soluție alternativă la problema în discuție este oferită de dezvoltarea rețelelor de nouă generație (Next generation Networks - NGN), capabile să folosească mai multe tehnologii de transmisie într-un mediu multi-operator și să asigure mobilitate generalizată. Rețelele eterogene pot oferi conectivitate fiabilă și permanentă, satisfăcând și cerinţele serviciilor de comunicaţii caracteristice Orașelor Inteligente și Sistemelor de Transport Inteligente [1].

PROIECTAREA ARHITECTURII PLATFORMEI DE CONECTIVITATE

Proiectarea arhitecturii platformei de conectivitate poate fi realizată în mai multe etape plecând de la arhitectura de sistem prezentată în figura 1. Arhitectura de sistem include patru entităţi de bază : *Router*-ul Mobil Inteligent (Smart Mobile Router - SMR), instalat în autobuz, Gateway-ul (Service Continuity Gateway - SCG), Serverul de Suport (Ubiquitous Continuity Support Server - UCSS) și o bază de date centrală (Central Database - CD) care stochează informaţiile de stare ale reţelei eterogene (Network State Information - NSI). Entităţile SCG, UCSS și CD formează o Platformă Server de Aplicaţie (Application Server Platform - ASP), practic o platformă centralizată de suport pentru realizarea conectivităţii permanente. ASP are rolul de a aduna, procesa și stoca într-o bază de date centrală NSI achiziţionată de la rutele SMR. Această baze de date poate fi utilizată pentru a optimiza decizia *router*-elor SMR în selecţia reţelelor utilizate pentru transmisie.

SMR este echipat cu unul sau mai multe interfețe de rețea (radio și prin cablu), care permit conectarea terminalelor pasagerilor (*smartphone*-uri, *laptop*-uri, etc.), precum şi a diferitelor terminale ale vehiculului (senzori, com-

puter de bord, cameră de supraveghere, etc.). SMR este echipat de asemenea cu mai multe interfeţe celulare (3G/4G), şi interfeţe WLAN/WiFi folosite pentru a stabili conexiuni cu reţelele publice asigurând conectivitatea la Internet a terminalelor din autobuz.

SCG asigură continuitatea serviciului, atunci când traficul de date este rutat de la o conexiune *wireless* la alta. Soluția de conectivitate propusă se bazează pe cuplarea deschisă ("open-coupling") a rețelelor *wireless*, adică aceste rețele nu schimbă informații de stare și de management. *Router*-ele SMR și SCG comunică prin tuneluri virtuale stabilite în rețeaua radio eterogenă (vezi *Figura* 2). Această abordare permite o implementare rapidă și eficientă a soluției, asigurând independența față de operatorii de rețea. Pe de altă parte, *router*-ele SMR nu au acces la NSI disponibil în rețelele de transport ale operatorilor și trebuie să realizeze măsurarea parametrilor rețelei eterogene. Datele de semnalizare și de măsurători active sunt transmise prin canalele de date, ceea ce scade eficiența sistemului.

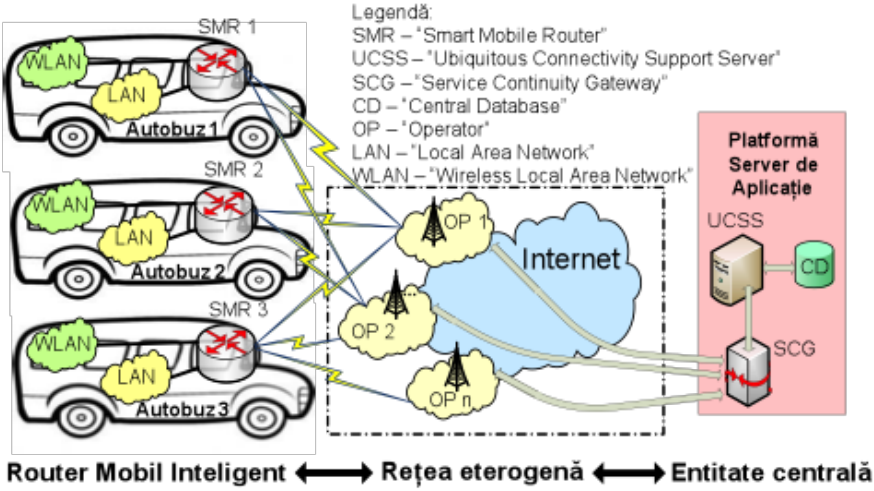


Figura 1. Arhitectura de sistem a platformei de conectivitate

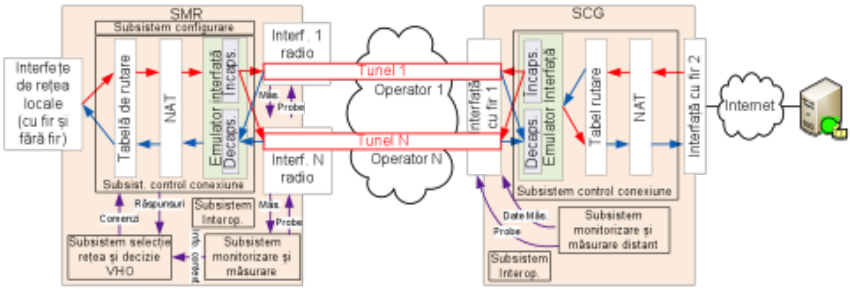


Figura 2. Soluția de conectivitate permanentă. Arhitectura funcțională a sistemului

A doua etapă în proiectarea arhitecturii constă în analiza funcțională a operațiunilor efectuate de către SMR obţinându-se arhitectura funcțională așa cum este prezentată în *Figura* 2. Subsistemele arhitecturii funcționale a SMR pot fi organizate în următoarele categorii: decizie, execuție, monitorizare, comunicare și configurare. Fiecare dintre aceste subsisteme include una sau mai multe module funcționale și sub-module. Modulul de decizie este reprezentat de Subsistemul de Selecţie a Reţelei și Decizie de Handover Vertical (Vertical Handover - VHO). Acest subsistem decide momentul inițierii procesului de *handover* și selectează rețeaua țintă pe baza informaţiei de stare a reţelei obţinută de la Subsistemul de Monitorizare. Entitatea de execuție este reprezentată de Subsistemul de Control al Conexiunii, care creează și controlează tunelurile IP stabilite peste reţelele disponibile și efectuează operațiunile de *handover* pe baza deciziilor primite de la Subsistemul de Decizie. Subsistemul include mai multe sub-module cum ar fi Tabelul de rutare, modulul de control NAT (Network Address Translation), Încapsulare-Decapsulare. Modulul de monitorizare este reprezentat de Subsistemul de Monitorizare și Măsurare, care include toate sub-modulele pentru achiziționarea informaţiilor de stare a reţelelor, a informațiilor legate de trafic și stocarea acestora într-o bază de date locală, parte a Subsistemului de Monitorizare. Modulul de interoperabilitate este reprezentat de Subsistemul de Interoperabilitate care asigură transmiterea mesajelor între subsistemele locale și cele distante. Acest subsistem, care acționează ca un comutator de mesaje, permite modularizarea arhitecturii SMR și o integrare ușoară a subsistemelor care efectuează diferite operațiuni. Subsistemul de Configurare permite configurarea interfeţelor radio, precizarea politicilor de control ale operațiunilor de *handover* și stabilirea cerințelor QoS ale serviciilor oferite

pasagerilor.

Arhitectura funcțională a SCG este similară cu cea a arhitecturii SMR-lui, principala diferență constând în lipsa subsistemului de decizie. Nicio decizie nu este luată de către SCG în ceea ce privește operațiunile de *handover*, iar Subsistemul de Monitorizare a reţelei are o structură simplificată, acesta oferind doar un sprijin pentru SMR în efectuarea măsurătorilor active.

Arhitectura funcțională a UCSS este mai simplă și include Subsistemul de Management al NSI. Această entitate oferă suport pentru SMR în luarea deciziilor de *handover*. Este important de menționat că deciziile de *handover* pot fi luate şi fără intervenția UCSS, doar pe baza măsurătorilor locale.

A treia etapă de proiectare a platformei constă în asocierea entităţilor arhitecturii funcţionale cu arhitectura platformei *hardware* utilizate. SMR a fost implementat pe un microcalculator care are performanțele necesare pentru efectuarea operațiunilor solicitate de sistemul de conectivitate. *Figura* 3 prezintă arhitectura platformei *hardware* care este echipat cu mai multe interfețe radio 3G/4G şi WiFi. Platforma este echipată și cu o unitate GPS care permite poziţionarea SMR în aria de acoperire a reţelei eterogene. Sistemul de operare care rulează pe microcalculator este un sistem de operare Linux minimal, ce include numai bibliotecile necesare pentru operațiunile efectuate de către SMR. *Figura* 3 prezintă de asemenea detalii despre comunicaţia dintre modulele arhitecturii funcţionale a SMR.

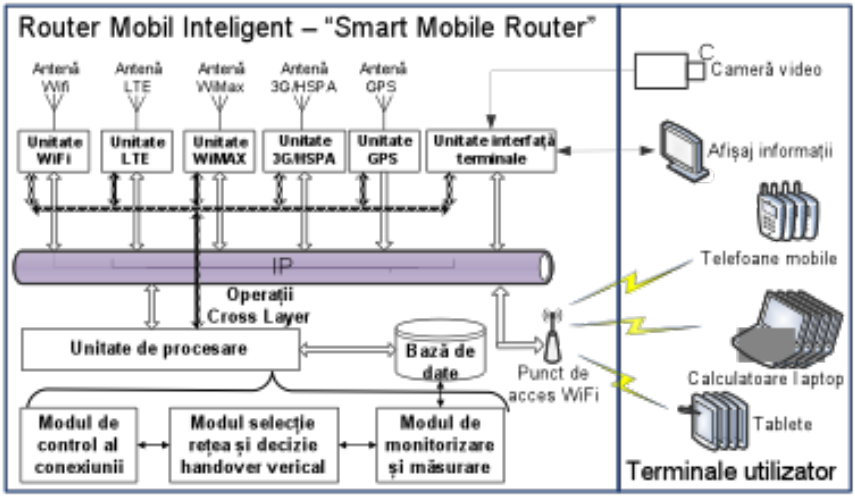


Figura 3. Arhitectura platformei hardware şi maparea modulelor funcţionale

PRINCIPII DE PROIECTARE ȘI IMPLEMENTARE A MECANISMELOR

Asigurarea conectivității permanente în rețele eterogene multi-operator și multi-tehnologie necesită o serie de mecanisme specifice, care trebuie să îndeplinească și anumite cerințe de flexibilitate și adaptabilitate la schimbările parametrilor rețelei eterogene.

MECANISMUL DE INTEROPERABILITATE

Acest mecanism, parte a Subsistemului de Interoperabilitate, permite schimbul flexibil de mesaje între procesele care colaborează pentru asigurarea conectivității omniprezente transmițând mai departe atât mesajele destinate proceselor locale cât și a celor distante. Procesele locale sau distante, care schimbă mesaje trebuie să fie conectate la un Modul de Interoperabilitate local, respectiv la unul similar care rulează pe dispozitivul distant. Mecanismul de interoperabilitate permite exploatarea deplină a caracterului modular al arhitecturii platformei, fiind posibilă înlocuirea sau actualizarea mecanismelor și a algoritmilor, fără a fi necesară schimbarea mecanismelor care interacționează cu cele în curs de actualizare. Fiecare modul *software* care comunică cu alte module trebuie să aibă un ID unic în cadrul platformei.

După ce un modul *software* se conectează la modulul de interoperabilitate, prima operație efectuată este de a trimite ID-ul său unic. Din acel moment, orice alt proces poate schimba mesaje cu cea actuală. Mesajele schimbate între procesele trebuie să aibă următorul format: <dest IP>,<ID dest>,<src IP>,<src ID>,<Mesaj>

MECANISMUL DE ACHIZIȚIE A INFORMAȚIEI DE STARE A REȚELEI

Acest mecanism, parte a Subsistemului de Monitorizare și Măsurare, achiziționează parametrii de stare ai rețelei eterogene prin efectuarea unor operații de monitorizare pasivă și prin măsurători active. Informația de stare achiziționată este procesată și apoi stocată într-o bază de date locală. Subsistemul de monitorizare are o structură modulară, un modul separat fiind dedicat fiecărui tip de interfață radio (WiFi, 3G/HSPA, 4G/LTE, WiMAX, etc.). În acest fel extinderea platformei cu interfețe de rețea pentru noi teh-

nologii de acces se poate face ușor și fără să fie afectată monitorizarea tehnologiilor de acces deja integrate în platformă.

Informațiile de stare sunt grupate în trei categorii principale: parametrii legăturii radio (ex. puterea semnalului, SINR, etc.), parametrii de trafic (ex. numărul de pachete primite/transmise, clasa de trafic, debitul instantaneu sau cel mediu, etc.) și parametrii de rețea (ex. tipul rețelei, clasa de serviciu, numele operatorului, etc.). Această clasificare a parametrilor rețelei permite o procesare mai flexibilă în condițiile în care se integrează în platformă noi tehnologii de acces radio.

Mecanismul de monitorizarea parametrilor rețelelor WiFi se bazează în special pe funcțiile oferite de instrumentele integrate în sistemul de operare Linux (*iw tools*), care rulează pe platforma de conectivitate. Sistemul de operare Linux oferă, de asemenea, un sistem de fișiere virtual, */proc/net/wireless*, care stochează mai mulți parametri ai conexiunilor WiFi active. Folosirea unor seturi specific de funcții oferite de programe Daemon (ex. *WPA supplicant*), reprezintă o altă alternativă pentru monitorizarea WiFi.

Mecanismul de monitorizare a interfețelor de rețea celulară poate fi implementată prin utilizarea comenzilor AT specifice interfeței sau a unor protocoale de interfațare a echipamentelor modem cu terminalele de date, cum este protocolul QMI (Qualcomm Station Mobile Modem Interface).

MECANISMUL DE MONITORIZARE ȘI ANALIZĂ A TRAFICULUI

Mecanismul este parte a Subsistemului de Monitorizare și Măsurare, și are rolul de a identifica fluxurile de date care tranzitează *router*-ul SMR precum și de a măsura parametrii acestor fluxuri. Informațiile generate de acest mecanism sunt utilizate pentru implementarea unor operații de *handover* și agregare de bandă mai complexe și mai eficiente. Principalele operații efectuate de acest mecanism sunt:

- Identificare flux: un flux aplicație este clasificat pe baza antetului IP al pachetelor care ajung la SMR, antet care include cicni câmpuri: adresa IP sursă și destinație, identificatorul protocolului de transport, portul sursă și destinație.
- Monitorizare trafic în timp real: fiecare flux este împărțit în două fluxuri simplex și numărul de pachete/octeți trimiși în fiecare direcție este contorizat fiind calculat periodic debitul instantaneu și cel mediu.

- Selecția fluxurilor: majoritatea fluxurilor care traversează *router*-ul SMR sunt de scurtă durată (ex. trafic web, interogări DNS, cererile ARP, etc.). Deoarece metoda de identificare a fluxurilor este precisă, numărul fluxurilor detectate crește foarte rapid și este foarte dinamic. Deși este posibilă utilizarea în procesul de decizie a statisticilor colectate de la toate fluxurile detectate, este mai practic să fie luate în considerare numai fluxurile relevante, adică cele care au o durată minimă și un debit mai mare decât un prag minim impus. Fluxurile scurte în rafală sunt grupate într-un flux unic care este gestionat ca și oricare flux relevant. În acest fel complexitatea proceselor de decizie și execuție *handover* este păstrată la un nivel acceptabil.

MECANISMUL DE DECIZIE

Arhitectura modulară a platformei permite integrarea mecanismelor de decizie cu complexitate și performanțe diferite. În funcție de puterea de procesare și memoria instalată a platformei hardware se pot implementa mecanisme de decizie cu complexitate redusă care realizează o "simplă" operație VHO sau mecanisme de decizie cu complexitate mare capabile de agregarea lărgimii de bandă disponibilă într-o rețea eterogenă. Pentru a realiza această flexibilitate mecanismele de decizie trebuie să fie separate de cele de monitorizare și cele de rutare. Mecanismul de decizie trebuie să lucreze pe baza unui număr limitat de parametri de rețea și de trafic stocați în baza de date locală a SMR-ului de către mecanismele de monitorizare. Parametrii utilizați trebuie să fie atent selecționați pentru a fi posibilă luarea unor decizii rapide și cât mai apropiate de cea optimală. Drept consecință, schimbarea interfețelor radio sau ale API-urilor care permit accesul la aceste interfețe nu are nici un efect asupra procesului de decizie. Pe de altă parte schimbarea mecanismului de decizie nu necesită modificarea mecanismelor de monitorizare și de rutare. Mecanismul de decizie este unul bazat pe evenimente, ceea ce înseamnă că acesta reacționează la evenimente sesizate de mecanismele de monitorizare a rețelei și a traficului sau la evenimente legate de expirarea unor temporizatoare, ceea ce reduce puterea de procesare necesară. Comenzile generate de procesul de decizie stabilesc pentru fiecare flux de date identificat care este rețeaua operatorului care trebuie folosită pentru transmiterea acestui flux și aceste comenzi sunt trimise mecanismului de control al conexiunii, care efectuează operațiile de rutare necesare.

MECANISMUL DE CONTROL AL CONEXIUNII

În funcție de complexitatea procesului de decizie și de cuplajul dintre rețelele operatorilor pot fi utilizate diferite mecanisme de management al mobilității și control al conexiunii, cum ar fi mecanismul *Mobile IP*, mecanisme bazate pe protocolul *SIP* sau mecanisme bazate pe instrumentele *iproute*. O soluție alternativă, integrată în arhitectura platformei și testată în condiții reale, este reprezentată de utilizarea tuneluri virtuale *VPN* (Virtual Private Networks) create între fiecare *router* SMR și *gateway*-ul SCG, peste toate rețelele de acces radio disponibile. Acest mecanism rutează fluxurile de date pe tuneluri pe baza comenzilor primite de la mecanismul de decizie. Tunelurile virtuale sunt create cu ajutorul *software*-ului OpenVPN, iar rutarea peste aceste tuneluri este implementată prin utilizarea mai multor tabele de rutare virtuale, unul pentru fiecare conexiune disponibilă. Pentru a ruta fluxurile de date pe diferite tuneluri se precizează care tabelă de rutare virtuală trebuie utilizată pentru fiecare flux în parte. Această operație se poate realiza cu instrumentele disponibile în biblioteca de funcții *iptables* integrate în sistemul de operare Linux. Cu ajutorul acestor instrumente se poate atașa un marcaj pentru orice pachet care are un set specific de proprietăți (ex. adresă IP sursă și destinație, port sursă și destinație și protocol de transport utilizat). Întregul proces trebuie să fie efectuate atât pe SMR și SCG, astfel încât setările să fie consistente pe ambele dispozitive.

EVALUAREA ARHITECTURII PLATFORMEI ŞI A MECANISMELOR DE CONECTIVITATE

Conceptele şi principiile de arhitectură precum şi principiile de proiectare ale mecanismelor de conectivitate expuse au fost utilizate pentru implementarea unei platforme de conectivitate în cadrul proiectului de cercetare FP7-UCONNECT, finanțat de Comisia Europeană. Această platformă a fost testată în medii de transmisie eterogene compuse din reţele WiFi şi celulare 3G/HSPA atât în condiţii de mobilitate redusă cât şi în condiţii de mobilitate vehiculară. Rezultatele acestor teste au validat pe deplin conceptele de arhitectură cât şi proiectarea şi implementarea mecanismelor de conectivitate. Rezultatele testelor au fost raportate în mai multe publicaţii ştiinţifice [2] [3] [4] [5] și rapoartele interne ale proiectului FP7-UCONNECT.

CONCLUZII

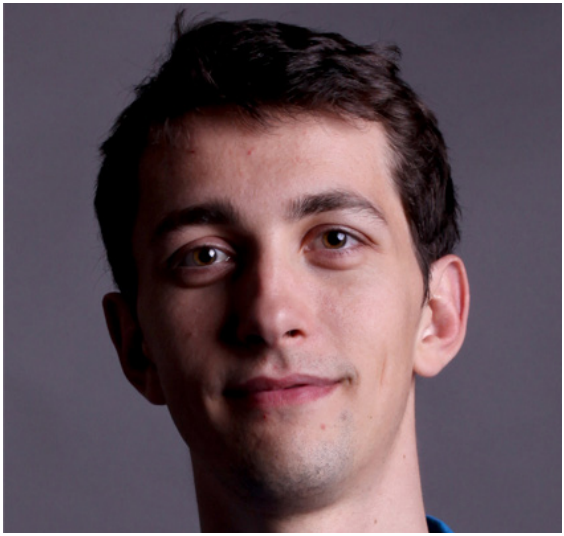
În contextul oferirii de conectivitate radio vehiculelor, au fost dezvoltate și standardizate diferite sisteme de comunicaţii radio. Un astfel de exemplu este sistemul CALM [6] dezvoltat pentru Sisteme de Transport Inteligente. Arhitectura sistemului este concepută pe baza funcționalităților oferite de stiva de protocoale TCP/IP și oferă conectivitate multi-tehnologie, dar nu definește mecanismul de execuţie al procesului de *handover* vertical. În acest caz procesul VHO se realizează prin protocolul Mobile IPv6 și, prin urmare, este nevoie de un grad ridicat de interoperabilitate a rețelelor. În comparație sistemul CALM arhitectura prezentată în această lucrare integrează mecanismele de decizie și de execuție VHO, ceea ce permite oferirea de conectivitate atât în rețelele IPv4 cât și IPv6 fără a utiliza funcţionalităţi oferite de operatori. Sistemul prezentat permite de asemenea și funcţionalităţi suplimentare, cum ar fi agregarea de bandă de la mai multe tehnologii de acces fără fir, ceea ce asigură conectivitate la Internet chiar și în zonele cu acoperire slabă sau în reţele supraîncărcate.

REFERINȚE

1. [1] UCONNECT FP7 project website, http://idi.gowex.com/uconnect/index.html
2. [2] Z. A. Polgar, A. B. Rus, Z. I. Kiss, A. Consoli, J. Ayadi, M. Egido, "Ubiquitous Connectivity Platform for Public Transport Communication Services", Future Network and Mobile Summit 2013, 3-5 July 2013, Lisboa, Portugal
3. [3] A. C. Hosu, Z. I. Kiss, Z. A. Polgar, "A cellular - WLAN vertical handover management system for public transport", 36th International Conference on Telecommunications and Signal Processing 2013 TSP 2013, 2-4 July 2013, Rome, Italy
4. [4] Z. I. Kiss, A. B. Rus, V. Dobrota, A. Consoli, M. Egido, Z. A. Polgar, "Seamless Connectivity Platform Architecture for Public Transportation", 21st International Conference on Software, telecommunications and Computer Networks 2013, 18-20 September 2013, Primosten, Croatia.
5. [5] A. C. Hosu, Z. I. Kiss, I. A. Ivanciu, Z. A. Polgar, A. Consoli, M. Egido, "Ubiquitous Connectivity Platform for Intelligent Public Transportation Systems", 10th ITS European Congress 2014, 16-19 June 2014, Helsinki, Finland.
6. [6] ISO 21210:2012: "Intelligent transport systems-Communications access for land mobiles (CALM)-IPv6 Networking", http://www.iso.org/iso/

Autori

Zsolt Alfred Polgar, Andrei Ciprian Hosu, Zsuzsanna Ilona Kiss, Andrei Bogdan Rus, Gabriel Lazăr, Virgil Dobrotă
@ Departamentul Comunicații, Universitatea Tehnică din Cluj Napoca



Șerban Meza

Activând la Universitatea Tehnică din Cluj-Napoca, în cadrul Centrului de Tehnologii Multimedia și Educației la Distanță, Șerban își pune în practică pasiunea pentru sistemele video și multimedia în activitățile de cercetare și integrare a acestora în educație și industrie. A fost activ implicat în proiectele educaționale românești www.didatec.ro, www.e-start.ro, www.e-administratie.forhe.ro si a colaborat cu compania Grass Valley – Olanda la realizarea camerelor video profesionale.

Actualmente este interesat de platforme educaționale social – interactive și de sistemele video 3D de realitate augmentată.

12

E-EDUCAȚIE ȘI PLATFORME EDUCAȚIONALE ONLINE

MULTIMEDIA SYSTEMS AND APPLICATIONS LABORATOR

MSAL, anterior *Centrul de Tehnologii Multimedia și Educație la Distanță (CTMED)*, funcționează în cadrul Universității Tehnice din Cluj-Napoca și este dedicat atât activităților de cercetare aplicativă în domeniu cât și implementării de soluții practice având la bază tehnologii multimedia. Astfel, pe lista subiectelor de interes și în care sunt deținute competențe se află:

- sisteme de e-educație, *blended learning* și management educațional.
- e-Sănătate: platforme și aplicații specifice de management sau de prelucări și procesări de informație medicală (HIS, RIS, HL7, DII-COM, imagistică medicală).
- e-Mediu și platforme și sisteme de management inteligent al clădirilor.
- e-Guvernare și administrare.
- dezvoltarea de conținut digital interactiv, multimedia: foto, grafică 2D și 3D, video 2D, 3D, 360, IR, realitate augmentată și virtuală.
- tehnici și tehnologii de reprezentare, codare și compresie a informațiilor multimedia și implementarea de standarde audio-video.
- scriere și management / implementare de proiect.

Proiectele majore în zona educațională aflate în ultimii ani în implementare în cadrul colectivului sunt:

- **DidaTec** -*Școala universitară de formare inițială și continuă a personalului didactic și a trainerilor din domeniul specializărilor tehnice și ingi-*

nerești – finanțare din Fonduri structurale, coordonator proiect fiind Universitatea Tehnică din Cluj-Napoca.

- **eSTART** - *Program multi-regional de studii masterale în domeniul eActivități* – finanțare din Fonduri structurale, coordonator proiect fiind Universitatea Tehnică din Cluj-Napoca.
- **eAdministrație** – *Pregătirea sistemului național de e-Administrație în România* – finanțare din Fonduri structurale, coordonator proiect Unitatea Executivă pentru Finanțarea Invățământului Superior, a Cercetării, Dezvoltării și Inovării.

Proiectul ***DidaTec*** a avut ca scop implementarea eficientă a tehnologiilor și instrumentelor educaționale moderne în învățământul superior tehnic. Principalele rezultate au fost realizarea unui Program de Formare în Tehnologii Educaționale moderne dedicat cadrelor didactice din învățământul superior tehnic din România și rularea acestuia în regim *blended-learning* în rândul a peste 750 de persoane din grupul țintă. Competențele dezvoltate au ajutat la transpunerea în format electronic (multimedia) de către aceștia

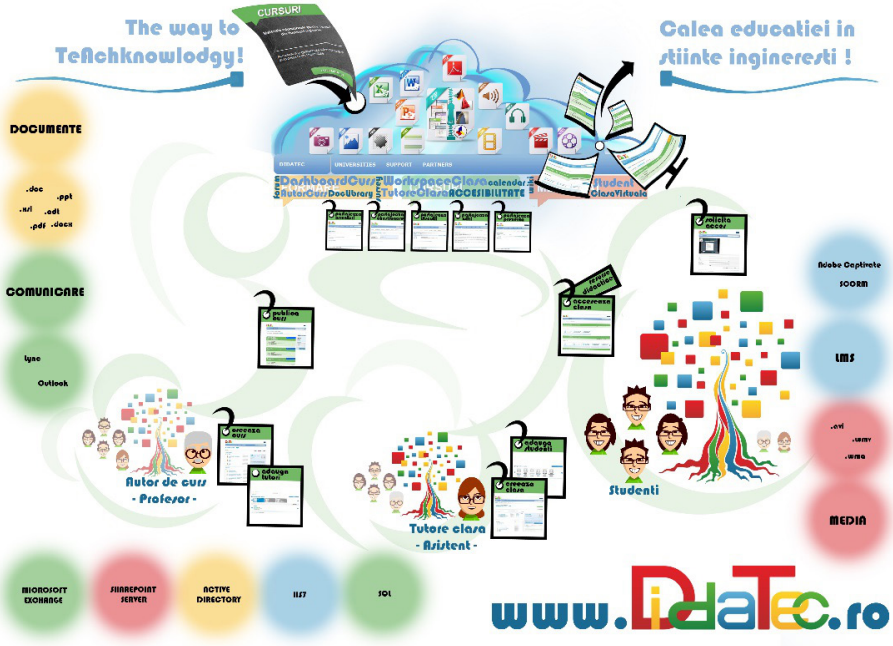


Fig. 1 Scenarii de utilizare a platformei www.didatec.ro

a materialelor deținute pentru disciplina predată în funcție de specificul existent. Programul de Formare și materialele create pentru cele peste 750 de discipline sunt accesibile prin intermediul platformei de management educațional dezvoltată în cadrul proiectului[1]. Tehnologia / platforma de implementare utilizată este MS SharePoint 2010. Imaginea următoare descrie principalele scenarii de utilizare.

Proiectul ***eSTART*** a avut drept scop realizarea unui program profesional de master derulat în regim de frecvență redusă și folosind platforme educaționale suport pe domeniul interdisciplinar al e-Activităților și e-Serviciilor cu specializările: Specialist platforme și servicii electronice în domeniul medical și de sănătate, Specialist platforme și servicii electronice în domeniul afacerilor (și bancar), Specialist platforme și servicii electronice în administrația publică, Specialist platforme și servicii electronice *new media*. În cadrul parteneriatului proiectului, în baza consultărilor cu partenerii sociali implicați (mediul economic, studenți, cadre didactice, etc.), s-a dezvoltat întregul curriculum necesar derulării programului de studii și s-a implementat o platformă de *e-learning* specifică. Rezultate disponibile în acest sens găsiți la adresa www.e-start.ro. Platforma de implementare utilizată este Moodle. Detalii generale legate de structura curriculumului sunt prezente în figura următoare:

Fig. 2 Structura curriculumului Programului de master eSTART

1 www.didatec.ro

Proiectul eAdministrație s-a adresat funcționarilor publici din România și a avut drept scop implementarea unei platforme educaționale *online* pentru pregătirea acestora în vederea dezvoltării competențelor legate de utilizarea tehnologiei informației în actul de guvernare și administrare. MSAL s-a implicat în definirea setului de cursuri oferite în cadrul proiectului precum și în implementarea și pilotarea a două dintre acestea: ”Conținut electronic pentru e-Administratie” și, respectiv, ”Proiectarea și implementarea fluxurilor de documente electronice”. Platforma cu materialele educaționale din proiect, implementată folosind platforma Ilias, este disponibilă la adresa http://learning.forhe.ro/.

Autor

dr.ing.ec. Șerban-Nicolae MEZA - Șef lucrări @ Universitatea Tehnică din Cluj-Napoca, în cadrul Centrului de Tehnologii Multimedia și Educației la Distanță

Silviu Dumitrescu

După cum probabil ştiţi sunt braşovean, fapt de care sunt foarte mândru.

Am două licenţe: în inginerie şi în matematică-informatică (pe vremea în care licenţa dura 5 ani), un masterat în informatică (jumătate făcut la Universitatea din Oldenburg) şi un doctorat în informatică, avându-l ca mentor pe dl. academician Solomon Marcus, un profesor şi un om absolut remarcabil.

Toată viaţa mea mi-am petrecut-o între nevoia de a studia, înțelege, inova şi dorinţa de a fi apropiat de tehnologie, de a fi productiv, de a arăta ceea ce am învăţat.

Acestea s-au concretizat în educarea multor generaţii de oameni cărora am încercat, şi într-o bună măsură am reuşit, să le transmit ideea de a nu se lăsa conduşi de interese mici, de moment. Pe de altă parte am proiectat şi dezvoltat aplicații bine vândute şi foarte apreciate, am scris cărţi şi articole.

Ca personalitate pot spune că sunt, aşa cum o persoană foarte apropiată declara, „complex", fac multe activităţi complementare care mă îndepărtează de viaţa monotonă, cotidiană. Nu vă spun ce carte am citit ultima oară, dar vă spun ultimul concert la care am fost: Lenny Kravitz, Munich, 15 noiembrie 2014 (încântător).

Îmi place să trăiesc printre oameni şi de foarte puţine ori am dezamăgit, iar când iubesc, arareori iubesc, o fac cu tot sufletul meu şi mult în plus.

13

JAVA STANDARD 8, NOUTĂŢI ŞI ÎMBUNĂTĂŢIRI

Cartea care însoţeste evenimentul „IT Days" din acest an, 2014, are titlul „Cum să construieşti un proiect IT". Deosebit de generos ca întindere şi ca subiecte posibil de abordat, titlul îşi găseşte şi o importantă latură tehnică, pe care o voi dezvolta din prisma noutăţilor de pe piaţa dezvoltarilor de soft ce folosesc lumea Java. Versiunea 8 a limbajului standard lansată anul acesta aduce numeroase noutăţi. Cred că pentru oricine este implicat în munca de dezvoltare a aplicaţiilor Java, o trecere în revistă a acestor noutăţi este binevenită, utilă şi interesantă.

Prima parte a articolului aduce în atenţie cele mai importante repere ale versiunii, cel puţin din perspectiva autorului. A doua parte este mult mai pragmatică şi vine în întâmpinarea nevoilor unei echipe de dezvoltare, care-şi doreşte să pună în valoare avantajele implementărilor predefinite ale *design pattern*-ilor, ale numeroasele API-uri cu funcţii grafice, în a crea clienţi *rich* folosind Java într-un mod mai facil, mai productiv având consecinţe importante în creşterea calităţii soft-ului.

Java SE8 este considerată revoluţionară prin câteva dintre noile caracteristici introduse. Programată iniţial pentru luna septembrie 2013, lansarea a fost amânată pentru martie 2014. Motivele sunt numeroase, dar ţin în special de corectarea *bug*-urilor şi îmbunătăţirea securităţii, mai ales pe acelea legate de client, având ca principal motiv JavaFX.

LAMBDA EXPRESII ŞI PROCESAREA COLECŢIILOR

Multe sunt modificările şi adăugările făcute în limbaj, dar probabil cea mai spectaculoasă este introducerea capabilităţilor *lambda*. Acestea sunt văzute ca un important beneficiu în programarea paralelă. De fapt, eforturile pentru creşterea performanţei în programarea paralelă s-au văzut încă din versiunea 7, introducerea *framework*-ului Fork-Join este un exemplu.

O funcţie *lambda* (funcţie anonimă) este o funcţie definită şi apelată fără a fi legată de un identificator. Funcţiile *lambda* sunt o formă de funcţii încuibate (*nested functions*) în sensul că permit accesul la variabilele din domeniul funcţiei în care sunt conţinute.

Funcţiile anonime au fost introduse de către Alonzo Church în anul 1936, în teoria sa despre calculele *lambda* (http://en.wikipedia.org/wiki/Lambda_calculus). În limbajele de programare funcţiile anonime au fost implementate încă din anul 1958, ca parte a Lisp.

În limbajele orientate pe obiect, precum Java, apar concepte similare, precum clasele anonime. Totuși, Java abia în versiunea 8 implementează funcţiile anonime. Alte limbaje, C#, JavaScript, Perl, Python, Ruby, ofereau de mult suport pentru acest concept.

Lambda expresiile ne permit să creăm instanţe ale claselor cu o singură metodă, într-un mod mult mai compact.

O *lambda* expresie constă din:

- o listă de parametri formali, separaţi prin virgulă şi cuprinşi eventual între paranteze rotunde,
- o săgeată direcţională ->,
- un body ce constă dintr-o expresie sau un bloc de instrucţiuni.

O interfaţă funcţională (*functional interface*), anotată `@FunctionalInterface`, este orice interfaţă ce conţine doar o metodă. Deoarece interfaţa funcţională are o singură metodă, putem omite numele metodei la implementare şi putem astfel elimina folosirea claselor anonime, cu implicaţii în creşterea performanţei. În locul claselor anonime folosim *lambda* expresii.

Pentru a înţelege modul în care se lucrează cu *lambda* expresii am construit un mic exemplu prin care am creat colecţii de obiecte sortate după diverse criterii. Implementarea interfeţei `Comparator` a fost făcută într-o clasă anonimă, folosind *lambda* expresii. Implementarea cu *lambda* expresii a fost posibilă pentru că în versiunea 8 `Comparator` este anotată cu `@FunctionalInterface`.

Elementul de bază al colecţiei este clasa `Product`, care este o clasă POJO cu getter-i si setter-i. Clasa conţine două implementări anonime ale comparatorului, determinând sortarea crescătoare respectiv descrescătoare a elementelor colecţiei.

```
package model;

import java.util.Comparator;

public class Product {
    private String name;
    private int price;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public void printProduct() {
        System.out.println(this.toString());
    }

    @Override
    public String toString() {
        return "Product [name=" + name + ", price=" + price +
"]";
    }

    public static Comparator<Product> ascendingPrice = (p1, p2) ->
{
        return p1.getPrice() - p2.getPrice();
    };

    public static Comparator<Product> descendingPrice = (p1, p2)
-> {
```

```
                return p2.getPrice() - p1.getPrice();
        };
    }
```

Clasa de test va aduce ceva în plus faţă de o clasă folosită până în versiunea 8 și anume procesarea colecţiei, care nu se va face într-un *foreach* clasic. Ca parte a API-ului `Collections` avem noul API `java.util.stream` ce oferă suport pentru operaţii funcţionale pe *stream*-uri de elemente. În exemplul nostru vom folosi o interfaţă de bază a acestui API şi anume `Consumer`, care reprezintă o operaţie ce acceptă un singur argument de intrare şi nu returnează ceva. Cu `Consumer` vom putea folosi *lambda* expresii:

```
import java.util.Set;
import java.util.TreeSet;
import java.util.function.Consumer;

import model.Product;

public class TestLambda {

    public static void processProducts(Set<Product> products,
                Consumer<Product> block) {
            for (Product p : products) {
                block.accept(p);
            }
    }

    public static void main(String[] args) {

    Product p1 = new Product();
    p1.setName("onion");
    p1.setPrice(10);
    Product p2 = new Product();
    p2.setName("tomato");
    p2.setPrice(20);

    Set<Product> ascendingPriceProducts = new TreeSet<>(
                Product.ascendingPrice);
    ascendingPriceProducts.add(p1);
    ascendingPriceProducts.add(p2);

    System.out.println("In ascending order:");
    processProducts(ascendingPriceProducts, p -> p.printProduct());

    Set<Product> descendingPriceProducts = new TreeSet<>(
```

```
                    Product.descendingPrice);
descendingPriceProducts.add(p1);
descendingPriceProducts.add(p2);

System.out.println("\nIn descending order:");
processProducts(descendingPriceProducts, p -> p.printProduct());

  }
}
```

Ca urmare a folosirii API-ului, *stream*-operaţiile efectuate pe o colecţie, numite și operații agregat, pot fi mult mai complexe decât cele ilustrate în exemplu şi anume: filtrarea după un predicat de selecţie, maparea obiectului filtrat, respectiv executarea unei acţiuni pe fiecare obiect mapat. Eu am prezentat doar ultima operaţie.

Observaţia pe care vreau să o fac codului anterior este că implementarea comparatorului ţine loc de suprascriere a funcţiei `equals()`, fapt ce poate fi dovedit prin modificarea, în cod, la aceeași valoare a prețului.

ENGINE-UL JAVASCRIPT NASHORN

Pe lângă *lambda* expresii, o caracteristică importantă a platformei standard 8 este dezvoltarea *engine*-ului JavaScript Nashorn (se pronunţă nazhorn). Prin acesta se pot integra *script*-uri JavaScript în codul Java clasic. Acest *engine* se bazează pe standardul ECMAScript 262. Este un *engine* scris complet de la zero, având ca obiectiv creşterea performanţei. Este astfel complet diferit faţă de *engine*-ul deja existent Rhino.

Voi da doar un exemplu de folosire a acestui *engine*:

```
import javax.script.*;

public class EvalScript {
    public static void main(String[] args) throws Exception {

    ScriptEngineManager factory = new ScriptEngineManager();
    ScriptEngine engine = factory.getEngineByName("nashorn");

          try {
                engine.eval("print('Hello, World!');");
          } catch (final ScriptException se) {
                se.printStackTrace();
          }
    }
```

```
}
```

Rulând acest exemplu vom obține la consolă mesajul `Hello, World!`.

ELEMENTE AVANSATE

În secțiunea finală a primei părți aduc în discuție anumite elemente cu un nivel de dificultate mai ridicat, pentru a permite evidențierea unor aspecte de performanță, productivitate și de reducere a dimensiunii codului scris.

Pentru început revin la *lambda* expresii. Prin *lambda* expresii putem crea metode anonime. Uneori însă, *lambda* expresiile apelează metode care au deja un nume.

Am definit într-una dintre secțiunile anterioare o clasă `Product` cu două atribute: `name` și `price`, *getter*-i și *setter*-i. Voi mai adăuga proiectului nostru o clasă POJO în care se află diverse metode de comparare, cu semnături apropiate de ale metodei `compare()` din interfața `Comparator`:

```
public class ProductComparisons {
    public int compareByName(Product a, Product b) {
           return a.getName().compareTo(b.getName());
    }

    public int compareByPrice(Product a, Product b) {
           return a.getPrice() - b.getPrice();
    }
}
```

În clasa de test vom crea două obiecte `Product`, `p1` și `p2`, pe care le vom pune într-un *array*:

```
Product[] basket = { p1, p2 };
Vom sorta array-ul folosind lambda expresii:
ProductComparisons myComparison = new ProductComparisons();
Arrays.sort(basket, (a,b)->myComparison.compareByName(a, b));
```

Această sintaxă este posibilă pentru că rezultatul *lambda* expresiei este o clasă anotată `@FunctionalInterface`, în cazul nostru, `Comparator`.

În locul folosirii *lambda* expresiilor, Java SE8 oferă posibilitatea utilizării referințelor de metode, prin intermediul operatorului de domeniu `::`. Sintaxa este astfel mult simplificată.

Revin la exemplul nostru și aplic operatorul menționat. Sintaxa devine:

```
Arrays.sort(basket, myComparison::compareByName);
```

Avem următoarele tipuri de referinţe:

- La o metodă a unui obiect (exemplul anterior);
- La o metodă statică (exemplul anterior poate fi ușor customizat);
- La o metodă a unui obiect arbitrar de un tip particular;

```
Arrays.sort(stringArray, String::compareToIgnoreCase);
```

- La un constructor:

```
Supplier<Product> s = Product::new;
```

unde, `Supplier` este din `java.util.function.Supplier`.

Un alt subiect supus atenţiei este legat de interfeţe. Interfeţele conţineau pîna la aceasta versiune doar semnături de metode şi constante. Java 8 propune o abordare care să îmbunătăţească utilizabilitatea interfeţelor. Dacă adaugăm noi semnături în interfaţă, atunci clasele ce o implementează ar trebui rescrise. Pentru a evita procesul rescrierii s-au introdus metodele *default*. Pe lîngă semnături şi constante, interfeţele vor conţine astfel şi implementări.

Voi da un exemplu simplu care să evidenţieze noile abordări:

```
public interface MyInterface {
    void myClassicMethod();

    default void myDefaultMethod() {
           System.out.println("hello default!");
    }

    static void myStaticMethod(){
           System.out.println("hello static!");
    }
}
```

Respectiv clasa ce implementează interfaţa:

```
public class MyClass implements MyInterface {
    @Override
    public void myClassicMethod() {
           System.out.println("hello classic!");
```

```
    }
}
```

Ca test avem:

```
public static void main(String[] args) {
    MyInterface mine = new MyClass();
    mine.myClassicMethod();
    mine.myDefaultMethod();
    MyInterface.myStaticMethod();
}
```

Surprindem astfel cele trei comportamente:

- Cel predefinit, cu calificatorul `public abstract` (Ex: `myClassicMethod()`),
- Cel cu implementare *default*,
- Cel cu implementare `static default`.

Procesul de moştenire are suport şi pentru acest nou comportament. Astfel:

- Metodele *default* care nu sunt menţionate în interfaţa derivată moştenesc comportamentul *default* din interfaţa de bază.
- Metodele *default* redeclarate în interfaţa derivată devin abstracte.
- Metodele default redefinite în interfaţa derivată vor fi folosite suprascris.

Nu voi incheia acest articol fără să descriu câteva dintre API-urile folosite în discuţiile anterioare:

- `java.util.function` furnizează interfeţele funcţionale ce sunt returnate ca tip de către *lambda* expresii. Interfeţele funcționale reprezintă concepte abstracte precum funcţiile, acţiunile sau predicatele.
- `java.util.stream` furnizează clase pentru operaţii pe *stream*-uri de elemente. *Stream*-urile diferă de colecţii prin:
    - *stream*-ul nu este o structură de date, ci transformă o structură de date într-un *pipeline* de operaţii.
    - o operaţie pe un *stream* produce o operaţie, dar nu modifică sursa.
    - operatiile pe *stream*-uri sunt implementate *lazy*, ceea ce înseamnă că pot expune oportunităţi de optimizare.
    - *stream*-urile sunt practic infinite. Există şi operatii de scurt-cir-

cuitare care să producă ieşiri într-un timp finit (`limit()`,`findFirst()`).
- *stream*-urile sunt consumabile, adică elementele sunt vizitate o singură dată. Pentru o revizitare trebuie creat un alt *stream*.

- toate clasele *wrapper* (`Boolean`, `Integer`, etc.) au fost îmbunătăţite cu metode care folosesc *lambda* expresi şi referinţe de metode.

JAVAFX - INTRODUCERE

JavaFX este urmașa lui F3 (Form Follows Function) ce îl are ca părinte pe Chris Oliver. În 2010, Oracle a anunțat că dezvoltarea lui JavaFX Script language va fi întreruptă, în schimb aceasta se va porta pe Java, formând platforma JavaFX 2. Prin aceasta se puneau bazele ca JavaFX să devină cel mai important mediu pentru aplicații *rich client*.

API-ul JavaFX este rulat de un *engine* compus din subcomponente. Aceasta cuprinde noul engine grafic de înaltă performanță numit Prism, sistemul eficient de *windowing* numit Glass și un *engine* media.

Glass Windowing Toolkit este responsabil cu furnizarea unui serviciu nativ ce include gestiunea ferestrelor, a *timer*-elor și a suprafețelor. De asemenea, leagă platforma JavaFX de sistemul de operare nativ. Mai mult, Glass este responsabil de gestiunea cozii de evenimente. Dacă AWT își gestionează propria coadă de evenimente, Glass utilizează coada nativă a sistemului de operare. Glass Toolkit rulează în același fir ca și aplicația JavaFX, în timp ce în AWT, spre exemplu, se crea un fir de execuție paralel cu cel al Javei.

O aplicație *rich client* este o aplicație ce are o interfață care referă *backend*-ul fără a aglomera astfel interfața utilizator. JavaFX are un set complet de butoane, diagrame, tabele și *container*-e de *layout* pe care le folosim pentru a crea interfețe utilizator *rich*. În plus, putem folosi stiluri CSS. Toate componentele se conectează și afișează date din *backend*.

În general, o aplicație *rich-client* are următoarele caracteristici:
- Este o aplicație *stand alone* executabilă.
- Conține o interfață utilizator ce are controale sau formulare.
- Este descărcabilă din *desktop* sau *web*.
- Se conectează la o bază de date și un server de *backend*.
- Este independentă față de sistemul de operare.

O aplicație JavaFX este o aplicație Java de bază, ce permite *feature*-uri

JavaFX. Codul minim necesar pentru a rula o aplicație JavaFX constă din:

- O clasă ce extinde clasa abstractă `javafx.application.Application`;
- O metodă `main()` ce apelează metoda `launch()` și suprascrie metoda abstractă `start()`. Ca bună practică, apelul metodei `launch()` este singurul apel din `main()`;
- Un *stage* primar ce este vizibil ca argument al metodei `start()`.

Avem trei tipuri de aplicații JavaFX:

- Aplicații propriu- zise, ce folosesc sintaxa Java tradițională și API-ul JavaFX.
- Aplicații FXML. FXML se bazează pe XML și este folosit pentru a defini interfețe utilizator în aplicații JavaFX. Cu FXML vom defini *layout*-uri statice precum formulare, controale sau tabele. Putem construi de asemenea *layout*-uri dinamice prin includerea unui *script.*
- Aplicații *preloader*, folosite în procesul de *deployment*

```
import javafx.application.Application;
import javafx.stage.Stage;

public class JavaFXApplication extends Application {

    @Override
    public void start(Stage stage) {
           stage.show();
    }

    public static void main(String[] args) {
           launch(args);
    }
}
```

Un *stage* (`javafx.stage.Stage`) este un *container* GUI,nivel de top pentru toate obiectele grafice, iar un *scene* (`javafx.scene.Scene`) este *container*-ul de bază. *Stage*-ul primar este construit de platformă, dar pot fi construite și alte obiecte *stage* de către aplicație. Obiectele *stage* trebuie să fie construite și modificate în firul aplicației JavaFX.

Articolele individuale care se află în interiorul scenei grafice sunt numite noduri. Fiecare nod este clasificat ca fiind unul dintre:

- *Branch* sau părinte, ceea ce înseamnă că poate avea descendenți.

- Frunză.

Primul nod din arbore este numit rădacină și nu are părinte.

Scena grafică este așadar o structură arborescentă. API-ul JavaFX face ca interfața grafică să fie mai ușor de creat, mai ales când sunt implicate efecte vizuale complexe și transformări.

API-ul scenei grafice JavaFX este un *retained mode*, adică el gestionează un model intern al tuturor obiectelor grafice din aplicație. În orice moment el știe ce obiecte să afișeze, ce zone ale ecranului necesită redesenare și cum să fie *renderizat* în cel mai eficient mod. Aceasta reduce semnificativ cantitatea de cod necesar în aplicație.

Îmbunătățesc aplicația anterioară prin adăugarea unui buton cu text și al unui *container* de *layout*. Butonul va fi un nod frunză, iar `StackPane`-ul nod rădăcină.

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class JavaFXApplication extends Application {

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World!");
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

      @Override
      public void handle(ActionEvent event) {
            System.out.println("Hello World!");
            }
      });

       StackPane root = new StackPane();
```

```
        root.getChildren().add(btn);
        primaryStage.setScene(new Scene(root, 300, 250));
        primaryStage.show();
    }
}
```

Voi dezvolta o aplicație echivalentă folosind FXML. FXML oferă următoarele avantaje:

- Fiind bazat pe XML este familiar multor dezvoltatori, în special dezvoltatorilor *web* și celor care utilizează alte platforme RIA.
- FXML nu se compilează, de aceea modificările sunt mult mai uşor de făcut.
- Asigură o structură ușor vizibilă a aplicației.

Când creăm o aplicație JavaFX FXML avem de creat trei fișiere:

- Clasa *main* ce conține cod pentru crearea scenei aplicației, a *stage*-ului și pentru lansarea aplicației. Una dintre acțiunile importante este dată de secvența următoare:

```
Parent root=null;
    try {
    root = FXMLLoader.load(getClass().getResource("Sample.fxml"));
    } catch (IOException e) {
            e.printStackTrace();
    }
    stage.setScene(new Scene(root));
```

Metoda `FXMLLoader.load()` încarcă ierarhia de obiecte din fișierul de resurse `Sample.fxml` și o asignează variabilei numite `root`. Această clasă este Model-ul în *pattern*-ul MVC. Ultima parte a acțiunii este setarea scenei.

- Fișierul `Sample.fxml` este fișierul în care construim interfața utilizator :

```
<?xml version="1.0" encoding="UTF-8"?>„

<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>

<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320"
xmlns:fx="http://javafx.com/fxml" fx:controller="henleyclient.Sample">
```

```
            <children>
   <Button id="button" layoutX="126" layoutY="90" text="Click Me!"
onAction="#handleButtonAction" fx:id="button" />

   <Label id="label" layoutX="126" layoutY="120" minHeight="16"
minWidth="69" prefHeight="16" prefWidth="69" fx:id="label" />
            </children>
   </AnchorPane>
```

Acest fișier reprezintă View-ul în *pattern*-ul MVC. Nodul rădăcină este `AnchorPane`, iar nodurile descendenți sunt `Button` și `Label`.

- Fișierul `Sample.java` este fișierul *controller* al interfeței utilizator. Numele acestuia trebuie să fie identic cu cel al clasei view fxml.

```
import java.net.URL;
import java.util.ResourceBundle;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;

public class Sample implements Initializable {

    @FXML
    private Label label;

    @FXML
    private void handleButtonAction(ActionEvent event) {
        System.out.println("You clicked me!");
        label.setText("Hello World!");
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }
}
```

Aplicațiile din acest articol au fost dezvoltate folosind Eclipse Luna, ca proiecte Java Standard în care am inclus pachetul `lib\jfxrt.jar`.

CONCURENȚA ÎN JAVAFX

Pachetul `javafx.concurrent` gestionează codul multifir al interacţiunii cu UI-ul şi asigură ca această interacţiune să aibă loc în firul corect. Pachetul constă din interfaţa `Worker` şi două clase de bază: `Task` şi `Service`, ambele implementând interfaţa `Worker`.

Interfaţa `Worker` furnizează API-ul folosit de un „background worker" ce comunică cu UI-ul. Clasa `Task` este o implementare complet observabilă a clasei `java.util.concurrent.FutureTask` şi permite dezvoltatorilor să implementeze *task*-uri asincrone în aplicaţiile JavaFX. Clasa `Service` execută aceste task-uri.

Un `Worker` este aşadar un obiect ce lucrează într-un fir din *background*. Starea obiectului `Worker` este observabilă și utilizabilă din firele aplicaţiei JavaFX.

Ciclul de viaţă al lui `Worker` este definit astfel: când este creat obiectul `Worker` este în starea `READY`. După ce a fost programat pentru lucru, obiectul `Worker` tranzitează către starea `SCHEDULED`. După aceea, când obiectul `Worker` rulează, starea sa devine `RUNNING`.

Observaţie: chiar dacă obiectul `Worker` a pornit imediat, fără a fi programat, el tranzitează totuşi în starea `SCHEDULED` şi apoi în `RUNNING`.

Starea obiectului `Worker`, atunci când se execută cu succes, devine `SUCCEEDED`, iar proprietatea valoare va fi setată la rezultatul obiectului `Worker`. Altfel, dacă sunt aruncate excepţii pe timpul execuţiei obiectului `Worker`, starea sa devine `FAILED`, iar proprietatea excepţie este setată la tipul de excepţie apărut. În orice stare obiectul `Worker` poate fi întrerupt utilizând metoda `cancel()`, ceea ce trimite obiectul în starea `CANCELLED`.

Progresul înregistrat la rularea obiectului `Worker` poate fi obţinut prin trei proprietăţi diferite: `totalWork`, `workDone` şi `progress`.

Clasa `Task` poate fi pornită în unul dintre următoarele moduri (primele două ar fi preferabile):

- Folosind ExecutorService API: `ExcecutorService.submit(task);`
- Utilizând metoda `task.run();`
- Pornind un fir cu *task*-ul dat ca parametru: `new Thread(task).start();`

Task-urile sunt utilizate pentru a implementa logica de lucru într-un fir din *background*. Pentru început trebuie să extindem clasa `Task`, care va suprascrie metoda `call()`. Clasa `Task` moşteneşte clasa `java.utils.con-`

`current.FutureTask`, ce implementează interfaţa `Runnable`. De aceea, obiectul `Task` poate fi utilizat cu API-ul Executor şi poate fi trimis unui fir, ca parametru.

Putem apela obiectul `Task` direct prin `FutureTask.run()`, ceea ce ne permite să apelăm acest *task* dintr-un alt fir.

Vom crea o clasă `CounterTask` ce extinde clasa `Task`.

```
public class CounterTask extends Task<Void> {
    @Override
    public Void call() {
            final int max = 10000000;
            updateProgress(0, max);
            for (int i = 1; i <= max; i++) {
                    updateProgress(i, max);
            }
            return null;
    }
}
```

Metoda `call()` este invocată de firul din *background*, de aceea această metodă poate manipula stări ce sunt sigure a fi citite sau scrise dintr-un fir din *background*. Spre exemplu, manipularea scenei grafice active din metoda `call()` va arunca o *runtime exception*.

Pe de altă parte, clasa `Task` este destinată a fi utilizată cu aplicaţii JavaFX şi ne asigură că orice modificări ale proprietăţilor publice, notificări de eroare, manipulatoare de evenimente şi stări apar în firul aplicaţiei JavaFX. În interiorul metodei `call()` putem utiliza metodele: `updateProgress()`, `updateMessage()` şi `updateTitle()` pentru a updata valorile corespunzătoare proprietăţilor pe firul JavaFX.

În aplicaţie am creat o instanţă a clasei anterioare, numită `countTask` şi am executat-o printr-un: `ExecutorService` (`ExecutorService es = Executors.newSingleThreadExecutor();`):

```
@Override
public void handle(ActionEvent event) {
    System.out.println("Count Started");
    bar.progressProperty().bind(countTask.progressProperty());
    es.execute(countTask);
}
```

Clasa `Service` este destinată executării unui obiect `Task` dintr-unul sau mai multe fire. Metodele şi stările clasei `Service` trebuie accesate din firul aplicaţiei JavaFX. Această clasă ajută dezvoltatorii să implementeze o inter-

acţiune corectă între firele din *background* şi firul aplicaţiei JavaFX. Putem porni, opri, anula şi restarta un `Service`. Un `Service` poate rula un *task* mai mult decât o dată. Aşadar, un serviciu poate fi definit declarativ şi restartat la cerere.

Un `Service` poate fi executat în unul dintre următoarele modalităţi:

- De un obiect `Executor`, dacă este specificat pentru serviciul dat;
- De un fir *daemon*, dacă niciun `Executor` nu este specificat;
- De un executor *custom* precum `ThreadPoolExecutor;`

Exemplu de creare a unui *service custom* este dat în exemplul de mai jos:

```
public class CounterService extends Service<Void> {
    @Override
    protected Task<Void> createTask() {
           CounterTask ct = new CounterTask();
           return ct;
    }
}
```

Creez în aplicaţia JavaFX un:

```
CounterService (CounterService cs = new CounterService();)
```

şi pornesc firul astfel:

```
if (cs.getState() == State.READY) {
    cs.start();
}
```

DATA BINDING ÎN JAVAFX

Data binding-ul are rolul de a simplifica *task*-ul sincronizând *view*-ul cu datele din model. Legarea (*binding*-ul) observă listele sale de dependenţe pentru a detecta schimbări şi se *auto-updatează* dacă acestea au apărut. API-ul de *binding* furnizează un mod simplu de a crea legări pentru cele mai comune situaţii.

Binding-ul este aşadar un mecanism puternic pentru exprimarea relaţiilor directe dintre variabile. Când obiectele participă la legături, modificările efectuate unuia vor fi automat reflectate celuilalt. Spre exemplu, *binding*-ul poate fi utilizat în GUI pentru păstrarea automată a afişărilor sincronizate cu datele pe care le referă.

Binding-urile sunt asamblate din una sau mai multe surse numite de-

pendenţe.

În exemplul nostru anterior am folosit funcţia `bind()` pentru a lega *progress bar*-ul de `counterTask`.

Iată codul complet al aplicaţiei JavaFX:

```
public class CounterBarAppService extends Application {
    StackPane root = new StackPane();
    VBox mainBox = new VBox();
    ProgressBar bar = new ProgressBar(0.0);
    CounterService cs = new CounterService();

    @Override
    public void init() throws Exception {
            super.init();

            mainBox.setAlignment(Pos.CENTER);
            mainBox.setSpacing(10);

            Button btn = new Button();
            btn.setText("Count to Ten Million!");
            btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
            System.out.println("Count Started");
            bar.progressProperty().bind(cs.progressProperty());
                    if (cs.getState() == State.READY) {
                            cs.start();
                    }
            }
    });

    Button restartBtn = new Button();
    restartBtn.setText("Restart");
    restartBtn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent event) {
            System.out.println("Count Started");
            bar.progressProperty().bind(cs.progressProperty());
                    cs.restart();
            }
    });

    mainBox.getChildren().add(btn);
    mainBox.getChildren().add(restartBtn);
```

```
            mainBox.getChildren().add(bar);
            root.getChildren().add(mainBox);
        }

        @Override
        public void stop() throws Exception {
            super.stop();
        }

        public static void main(String[] args) {
            launch(args);
        }

        @Override
        public void start(Stage primaryStage) {
            primaryStage.setTitle("JavaFX Service Example");

            primaryStage.setScene(new Scene(root, 400, 250));
            primaryStage.show();
        }
    }
```

INSTRUMENTE GRAFICE DE AFIŞARE A COLECŢIILOR ÎN JAVAFX

Tabelele reprezintă unul dintre cele mai puternice instrumente folosite în JavaFX pentru afişarea datelor, suportând următoarele acţiuni:

- Reordonarea coloanelor afişate de către utilizator,
- Sortarea multiplă a coloanelor,
- Redimensionarea,
- *Factories* de celulă pentru customizarea conţinutului celulei.

Mai multe clase din JavaFX SDK API sunt folosite pentru reprezentarea datelor în formă tabelară. Dintre acestea cele mai importante sunt `TableView`, `TableColumn` şi `TableCell`. Putem popula un tabel dintr-un model de date şi îi putem aplica apoi un *cell factory*.

`TableView` are facilităţi ce includ:

- API-ul `TableColumn`:
  - Suport pentru *cell factories*, ce permite o customizare a conţinutului celulei în ambele stări: de afişare şi de editare;
  - Specificarea dimensiunilor: `minWidth`, `prefWidth`, `maxWidth`,
  - a coloanelor de dimensiune fixă;

    - Redimensionarea la rulare de către user ;
    - Reordonarea la rulare coloanelor de către user;
    - Suport pentru încuibarea coloanelor;
- Diferite politici de redimensionare care determină ce se întâmplă atunci când utilizatorul redimensionează coloanele;
- Suport pentru sortarea pe mai multe coloane prin apăsarea *header*-ului de coloană (apăsând tasta Shift în timp ce facem *click* pe *header*);

Minimal avem nevoie de clasele următoarele pentru a crea un tabel:

- `TableView<S>`, unde `S` este tipul obiectului ce conţine lista itemilor din `TableView;`
- `TableColumn<S, T>`, unde `S` este varianta tipului generic `TableView`, iar `T` este tipul conţinutului tuturor celulelor din acest `TableColumn`.

`TableView` este construit dintr-un număr de instanţe `TableColumn`. Fiecare `TableColumn` este responsabilă de afişarea şi editarea conţinutului unei coloane. În plus, `TableColumn` conţine proprietăţi pentru:

- Redimensionare,
- Vizibilitate,
- Afişarea textului de *header,*
- Afişarea coloanelor încuibate pe care le poate conţine,
- Afişarea unui meniu de context atunci când utilizatorul face *click* dreapta pe zona capului de coloană,
- Posibilităţi de sortare.

Când creăm o instanţă `TableColumn`, probabil cele mai importante proprietăţi de setat sunt textul (adică ce afişăm în capul de tabel al coloanei) şi *cell value factory* (utilizat pentru popularea celulelor individuale din coloană).

Clasa `TableCell<S,T>` reprezintă intersecţia unei linii cu o coloană în `TableView`. Conţine următoarele proprietăţi:

- `tableColumn`: instanţa `TableColumn` din spatele `TableCell`,
- `tableView`, `TableView`-ul asociat cu `TableCell`,
- `tableRow`, `TableRow`-ul în care `TableCell`-ul este plasat.

Pentru a identifica intersecţia `TableCell` conţine o proprietate de index. `Cell<T>` este utilizat pentru o celulă individuală într-un `TableView`.

Fiecare celulă este asociată unui singur item de date, reprezentat de proprietatea `item`. O celulă este responsabilă de *renderizarea* item-ului care rezidă în el, care este de obicei un text. O celulă permite customizarea printr-un *cell factory*.

API-ul `Cell` este utilizat pentru virtualizarea controalelor precum `ListView`, `TreeView` şi `TableView`. Un `Cell` este un control etichetat, folosit pentru a *renderiza* o unitate într-unul dintre controalele mai sus amintite. `Cell` este responsabil atât pentru afişare cât şi pentru editarea *itemului*. Pe lângă text, `Cell` poate fi reprezentată de alte controale precum `CheckBox`, `ChoiceBox` sau orice `Node` precum `HBox`, `GridPane` sau chiar `Rectangle`. Deoarece `ListView`, `TreeView`, `TableView` precum şi alte asemenea controale pot fi folosite pentru afişarea unei cantităţi mari de date, nu este practic să creăm un `Cell` pentru fiecare item din control. Fiecare celulă este reutilizată, ceea ce face acest control virtualizat.

Deoarece `Cell` este un `Control`, el are în spate un model. `Skin`-ul său este responsabil pentru definirea *look and layout*-ului, în timp ce `Behaviour` este responsabilă pentru manipularea evenimentelor şi utilizarea acelor informaţii conținute pentru a modifica starea. `Cell` este stilizat prin CSS ca orice alt control.

Pentru a specializa o celulă utilizată pentru un `TableView` trebuie să furnizăm o implementare a funcţiei *callback* `cellFactory()` definită pe `TableView`. *Cell factory* este apelată de platformă ori de câte ori o nouă celulă trebuie să fie creată. Implementarea unui *cell factory* este responsabilă pentru crearea unei instanţe `Cell` şi pentru configurarea necesară pentru ca `Cell` să reacţioneze la schimbările stării sale.

Cell factory este responsabilă de virtualizarea *skin*-urilor de *container* pentru a *renderiza* reprezentarea predefinită a unui item `Cell`. Spre exemplu, într-un `ListView` converteşte itemii la un `String` şi apelează `Labeled.setText(String)`. Dacă dorim să specializăm celula utilizată în `ListView` trebuie să furnizăm implementarea funcţiei *callback* definită pe `ListView`.

Cell factory este apelat de platformă ori de câte ori determină că o celula trebuie să fie creată.

Spre exemplu, un `ListView` are 10 milioane de itemi. Crearea tuturor celor 10 milioane este foarte costisitoare. Aşadar, implementarea *skin*-ului `ListView` va crea doar atâtea celule cât să umple spaţiul vizual. Dacă se redimensionează spaţiul vizual, sistemul va determina dacă este nevoie de crearea altor celule. În acest caz va apela `cellFactory()` (dacă există vre-

una) pentru a crea o implementare `Cell`. Dacă nu este furnizată niciuna, implementarea predefinită este utilizată.

`ObservableList` este modelul de date care stă la baza lui `TableView`. O instanţă `TableView` este definită astfel:

```
TableView<Person> table = new TableView<>();
```

Am definit astfel un tabel primar. Modelul de date este creat pe baza unui `ObservableList`. Îl putem seta direct în `TableView`. Spre exemplu:

```
ObservableList<Person> teamMembers = getTeamMembers();
table.setItems(teamMembers);
```

O dată setată lista de itemi `TableView` este automat *updatată* ori de câte ori lista `teamMembers` se modifică. Dacă lista de itemi este disponibilă înainte ca `TableView` să fie instanţiat este posibil să o trimitem direct în constructor.

Ceea ce mai trebuie să facem este să împărţim datele conţinute în model în una sau mai multe instanţe `TableColumn`. Pentru a crea un `TableView` cu două coloane, ce afişează `firstName` şi `lastName` vom folosi codul:

```
    public class DataTable extends Application {
        TableView<Person> table = new TableView<>();
        BorderPane root = new BorderPane();

        VBox mainBox = new VBox(); // Container for content

        final ObservableList<Person> teamMembers = FXCollections
                        .observableArrayList(new Person("Ion", "Tech"),
new Person("Petre","Petrescu"), new Person("Doru", "Dorescu"), new
Person(
                                        "Vasile", "Vasilescu"));

        @Override
        public void init() {

    Label centerLbl = new Label("Persons");
    centerLbl.setStyle("-fx-font-size:16pt; -fx-font-weight:bold;");

    table.setItems(teamMembers);

    TableColumn<Person, String> firstNameCol = new TableColumn<Person,
```

```
String>("First Name");
   firstNameCol .setCellValueFactory(new PropertyValueFactory<Person,
String>("firstName"));
   TableColumn<Person, String> lastNameCol = new TableColumn<Person,
String>("Last Name");
   lastNameCol.setCellValueFactory(new PropertyValueFactory<Person,
String>("lastName"));

   table.getColumns().setAll(firstNameCol, lastNameCol);
   mainBox.getChildren().add(centerLbl);
   mainBox.getChildren().add(table);

   }

       @Override
       public void start(Stage primaryStage) {
              primaryStage.setTitle("Data Table");

              root.setCenter(mainBox);
              primaryStage.setScene(new Scene(root, 400, 300));
              primaryStage.show();

       }

       public static void main(String[] args) {
              launch(args);
       }
   }
```

Prin aceasta am definit complet proprietățile minime cerute pentru a crea o instanță `TableView`. Nici o altă proprietate a clasei `Person` nu va fi afișată pentru că nu avem definite alte `TableColumn`.

Implementarea unui *cell factory* este responsabilă nu doar pentru crearea instanței `Cell`, dar și pentru configurarea acelei celule. În exemplul următor am creat o clasă `Callback` ce are ca atribute instanțe ale claselor `TextAlignment` și `Format` cu parametri:

- `S`, tipul generic al lui `TableView;`
- `T`, tipul conținutului în toate celulele lui `TableColumn`

```
public class FormattedTableCellFactory<S, T> implements
              Callback<TableColumn<S, T>, TableCell<S, T>> {
       private TextAlignment alignment;
       private Format format;

       public TextAlignment getAlignment() {
              return alignment;
```

```
}

public void setAlignment(TextAlignment alignment) {
	this.alignment = alignment;
}

public Format getFormat() {
	return format;
}

public void setFormat(Format format) {
	this.format = format;
}

@Override
public TableCell<S, T> call(TableColumn<S, T> p) {
	TableCell<S, T> cell = new TableCell() {
		@Override
	public void updateItem(Object item, boolean empty) {
			if (item == getItem())
				return;

			super.updateItem(item, empty);

			if (item == null) {
				super.setText(null);
				super.setGraphic(null);
			} else if (format != null) {
			super.setText(format.format(item));
			} else if (item instanceof Node) {
				super.setText(null);
				super.setGraphic((Node) item);
			} else {
				super.setText(item.toString());
				super.setGraphic(null);
			}
		}
	};
	cell.setTextAlignment(alignment);
	switch (alignment) {
	case CENTER:
		cell.setAlignment(Pos.CENTER);
		break;
	case RIGHT:
		cell.setAlignment(Pos.CENTER_RIGHT);
		break;
	default:
```

```
                    cell.setAlignment(Pos.CENTER_LEFT);
                    break;
                }
                return cell;
            }
        }
```

Respectiv:

```
    FormattedTableCellFactory<Person, String> xx = new FormattedTable-
CellFactory<>();
    xx.setAlignment(TextAlignment.CENTER);
    firstNameCol.setCellFactory(xx);
```

Beneficiile utilizării CSS-urilor pentru a seta stilul unui tabel constau în eficienţa de timp şi eficienţa de memorie, uşurinţa utilizării şi construirii bibliotecilor pentru celule, uşurinţa customizării formatării de afişare.

Utilizăm CSS-ul pentru a seta culorile celulei:

- Fiecare celulă poate fi stilizată direct din CSS. Spre exemplu, dacă dorim să schimbăm culoarea predefinită a *background*-ului fiecărei celule din `TableView` la alb, trebuie să utilizăm următorul CSS

```
    .table-cell {
-fx-padding: 3 3 3 3;
-fx-background-color: white;
    }
```

- Pentru a seta culorile unor celule selectate dintr-un `TableView`, la albastru, vom folosi următorul CSS:

```
    .table-cell:selected {
-fx-background-color: blue;
    }
```

Pentru ca `table-cell:selected` să funcţioneze trebuie să setăm `cellSelectionEnabled` la `true`.

Multe implementări de celulă extind `IndexedCell` în loc de `Cell`. Aceasta permite adăugarea altor două pseudoclase: `odd` si `even`. Cu acestea putem obţine colorarea alternată a liniilor prin intermediul unui CSS de forma:

```
.table-row-cell:odd{
    -fx-background-color:lightblue;
}
```

Ca sursă bibliografică pentru acest articol am folosit documentaţia *free* a platformei Java SE8 si JavaFX, parte dintre exemple sunt originale, altele adaptate. Le mulţumesc colegilor mei pentru suport, asistenţă şi *feedback*.

Va mulţumesc vouă cititorilor, pentru lectură şi aştept cu plăcere, ca întotdeauna, discuţiile cu cei interesaţi de mai multe detalii.

Autor

Silviu Dumitrescu - Line Manager @ Accesa

Peter Lawrey

Lui Peter Lawrey îi place să inspire dezvoltatorii pentru a-şi îmbunătăţi calitatea soluţiilor lor, să-şi proiecteze sistemele după principiile simplităţii şi performanţei şi să se bucure mai mult de munca lor, fiind creativi şi inovatori.

El are un blog binecunoscut, „Vanilla Java" care atrage 120K accesări de pagini pe lună, este al treilea pe StackOverflow.com pentru Java şi al doilea pentru concurenţă şi este dezvoltator principal al proiectului OpenHFT, care include suport pentru off heap memory, thread pinning și low latency persistence şi IPC (doar 100 nano-secunde).

13

CHRONICLE MAP AND YAHOO CLOUD SERVICE BENCHMARK

Yahoo Cloud Service Benchmark is a reasonably widely used benchmarking tool for testing key value stores for a significant number of key e.g 100 million, and a modest number of clients i.e. served from one machine.

In this article I look at how a test of 100 million * 1 KB key/values performed using Chronicle Map on a *single machine* with 128 GB memory, dual Intel E5-2650 v2 @ 2.60GHz, and six Samsung 840 EVO SSDs.

The 1 KB value consists of ten fields of 100 byte Strings. For a more optimal solution, primitive numbers would be a better choice. While the SSDs helped, the peak transfer rate was 700 MB/s which could be supported by two SATA SSD drives.

These benchmarks were performed using the latest version at the time of the report, Chronicle Map 2.0.6a-SNAPSHOT.

MICRO-SECOND WORLD

Something which confounds me when reading benchmarks about key-value stores is that they start with the premise that performance is really important. IMHO, about 90% of the time, performance is **not** the most important feature, provided you have sufficient performance.

These benchmark reports then continue to report times in milli-seconds, not *micro-seconds* and throughputs in the *tens of thousands* instead of the hundreds of thousands or *millions*. If performance really was that important, they would have built their products around performance, instead of the *useful features* they do support, like multi-key transactionality, quorum updates

and other features Chronicle Map doesn›t support, for *performance reasons*.

So how would a key-store built for performance look with YCSB?

THROUGHPUT MEASURES

The “50/50” tests 50% random reads and 50% random writes, the “95/5” tests 95% reads to 5% writes. It is expected that writes will be more expensive, and a higher percentage of reads results in higher throughputs.

Threads	50/50 read/update	95/5 read/update
1	122 K/s	262 K/s
2	235 K/s	496 K/s
4	339 K/s	910 K/s
8	565 K/s	1.010 M/s
15	973 K/s	1.445 M/s
30	816 K/s	1.787 M/s

LATENCIES

The following latencies are in micro-seconds, not milli-seconds.

Threads: 8	50/50 read	95/5 read	50/50 update	95/5 update
average	5.7 µs	4.9 µs	13 µs	12.9 µs
95th	15 µs	13 µs	27 µs	25 µs
99th	25 µs	30 µs	44 µs	47 µs
worst	52 ms	52 ms	52 ms	52 ms

Note: the benchmark is not designed to be GC free and creates some garbage. This is not particularly high and the benchmark itself uses only about 1/4 of CPU according to flight simulator, however it does impact the worst latencies.

CONCLUSION

Make sure the key-value store has the features you need, but if performance is critical, look for a solution designed for performance as this can be 100x faster than full featured products.

OTHER HIGH PERFORMANCE EXAMPLES

Aerospike benchmark - Single server benchmark with over 1 M TPS, sub-micro-second latencies. Uses smaller 100 byte records.

NuoDB benchmark - Supports transactions across a quorum. 24 nodes for 1 M TPS.

Oracle NoSQL benchmark - A couple of years old, uses a lot of threads, otherwise a good result.

VoltDB benchmark - Not tested to 1 M TPS, but promising. Latencies around 1-2 ms, report has 99th percentile latencies which others don't include.

ROOM FOR IMPROVEMENT

MongoDB driver benchmark - Has 1000s of micro-seconds instead of milli-seconds.

Cassandra, HBase, Redis - Shows you can get 1 million TPS if you use enough servers, 288 nodes for 1 M TPS.

Report including Elasticsearch - Report includes runtime in a "resource Austere Environment"

Hyperdex - Cover throughput only.

WhiteDB - Reports latencies in micro-seconds for 170 K records, and modest throughputs.

Benchmark including Aerospace - Reports

FOOTNOTE

Using smaller values helps, and we suggest trying to make values closer to 100 bytes. This is the result of the 95/5 workload B, using 10x10 byte fields, and 50 M entries as the Aerospike benchmark does. (30 clients)

[OVERALL], RunTime(ms), 60,669
[OVERALL], Throughput(ops/sec), **3,296,576**
[READ], Operations, 190002671
[READ], AverageLatency(us), **4.81**
[READ], MinLatency(us), 0
[READ], MaxLatency(us), 74864
[READ], 95thPercentileLatency(ms), 0.009
[READ], 99thPercentileLatency(ms), 0.014
[READ], Return=0, 115841209
[READ], Return=1, 74161462
[UPDATE], Operations, 9997309
[UPDATE], AverageLatency(us), **12.23**
[UPDATE], MinLatency(us), 1
[UPDATE], MaxLatency(us), 75015
[UPDATE], 95thPercentileLatency(ms), 0.017
[UPDATE], 99thPercentileLatency(ms), 0.028
[UPDATE], Return=0, 9997309

Autor

Peter Lawrey - CEO @ Higher Frequency Trading

Victor Ionescu

Victor este un consultant IT care lucrează în industria de dezvoltare SAP. Prima sa întâlnire cu lumea SAP a avut loc aproximativ acum 3 ani şi jumătate, când a decis să încerce acest domeniu, care la acea vreme îi era complet necunoscut. De atunci, el a fost implicat în mai multe proiecte de dezvoltare SAP, asumându-şi o gamă largă de sarcini care acoperă întregul ciclu de dezvoltare software, de la consilierea clienţilor şi specificaţia funcţională a produsului, până la design tehnic şi implementare.

În afară de aceste activităţi legate de proiecte, Victor a şi scris mai multe articole cu privire la produsele şi tehnologia SAP şi a susţinut cursuri de dezvoltaresoftware ABAP SAP pentru studenţii de la Informatică din Cluj-Napoca. În timpul său liber, îi place să analizeze pieţele financiare, aceasta reprezentând de asemenea un domeniu în a cărui cercetare este implicat în mod activ la Universitatea Tehnică din Cluj-Napoca.

14

TENDINŢE ÎN DEZVOLTAREA DE SOLUŢII ENTERPRISE

În domeniul *software*, termenul "enterprise" a fost folosit întotdeauna asociat ideii de complexitate. Prin definiție, o soluţie *enterprise* trebuie să ofere un spectru larg de funcţionalităţi, să fie configurabilă până la ultimul detaliu, să facă faţă în condiţii de utilizare masivă și să aibă multe alte caracteristici similare, ce fac ca în mod inevitabil astfel de soluţii să devină un soi de aplicaţii-mamut, care știu să facă orice, mai puţin să fie intuitive și ușor de utilizat.

Dar studiile au arătat că, deși multitudinea de funcţionalităţi este adesea considerată esenţială, în realitate scenariile complexe reprezintă mai puţin de 20% din volumul de muncă tipic al unui utilizator de *software enterpri*se. Prin urmare e de așteptat ca această situaţie să se schimbe în viitor, din ce în ce mai mulţi clienţi declarând că uşurinţa în utilizare a *software*-ului are un rol cel puţin la fel de important cu multitudinea de funcţii pe care acesta le aduce.

O cauză a acestei schimbări de mentalitate este probabil şi discrepanţa foarte mare dintre aplicaţiile pe care le folosim în viaţa personală (vezi aplicaţiile Google, Facebook, Twitter etc.), și experienţa *enterprise*, marcată în primul rând de interfeţele complexe şi implicit de o curbă de învăţare mult prea „lină".

Ca dovadă a faptului că această necesitate este cât se poate de reală, putem lua exemplul [1] companiei de cosmetice Avon. Aceasta a fost nevoită să stopeze în anul 2013 implementarea unei noi soluţii de gestiune a comenzilor – este vorba de un proiect cu o valoare de 125 milioane USD- datorită faptului că angajaţii au început să demisioneze , considerând că noua soluţie este prea dificil de utilizat şi le îngreunează munca de zi cu zi.

Alt indicator al schimbării sunt diversele iniţiative implementate de către companii, iniţiative precum BYOD („Bring your own device") [2], prin care

se permite angajaţilor să folosească terminalele personale (telefon, tabletă etc.) în interes de serviciu. Această iniţiativă merge mână în mână cu ideea de mobilitate. De asemenea, un trend pe care companiile mizează din ce în ce mai mult din dorinţa de a eficientiza procesele interne, facilitând accesul angajaţilor la resursele interne ale companiei, indiferent unde s-ar afla aceștia.

Prin urmare, este evident faptul că şi soluţiile *enterprise* trebuie regândite, astfel încât să ţină pasul cu noile trenduri, oferind utilizatorilor un alt fel de experienţă, prin aplicaţii simple și intuitive. Însă tranziţia către o astfel de experienţă nu este una care să poată fi făcută cu ușurinţă, pentru asta fiind necesară o schimbare fundamentală a modului în care aceste aplicaţii sunt concepute.

Un prim pas ar putea fi renunţarea la genul de aplicaţii-mamut, gândite pentru a acoperi întregul spectru de funcţionalităţi, în favoarea unui set de aplicaţii, în care fiecare dintre acestea este concepută pentru un singur Use-Case. Evident aplicaţiile complexe nu pot să dispară întru totul de pe piaţă, în continuare existând situaţii (cei 20% menţionaţi anterior) în care este necesar un control mai amănunţit asupra sistemului.

Pentru restul situaţiilor însă pot fi dezvoltate astfel de mini-aplicaţii, care se remarcă prin avantajul de a avea un scop foarte bine definit, putând prin urmare „ghida" utilizatorul către atingerea acestuia, prin interfeţe intuitive, concepute special pentru acea situaţie.

Pe lângă ușurinţa în utilizare, nu trebuie neglijat nici faptul că experienţa ne-a arătat că, atunci când gândim la scară larga, cea de-a doua variantă (N funcţii - N aplicaţii) este mai ușor de implementat, întreţinut și extins decât varianta cu N funcţii înglobate într-o singură aplicaţie.

Prin astfel de demersuri practic putem spune că accentul se mută de pe multitudinea pe functionalităţi pe care o singură aplicaţie le poate îndeplini, pe capacitatea de integrare a multiplelor aplicaţii.

Însă adevărata provocare ţine de modul în care poate fi făcută din punct de vedere tehnic această tranziţie, astfel încât soluţiile business care și-au dovedit capacitatea în mediul productiv să poată fi folosite ca un nucleu al noii generaţii de *software enterprise*. Aceasta se întâmplă fiindcă în lumea reală clienţii sunt adesea refractari atunci când vine vorba de înlocuirea completă a soluţiilor existente cu unele noi. [3]

Iar în aceste situaţii de cele mai multe ori *gap*-ul tehnologic dintre sistemele „legacy" ale clienţilor și ceea ce numim „State-of-the Art" în materie de

Fig. 1 O nouă paradigmă de dezvoltare a aplicațiilor enterprise: de la N funcții - 1 aplicație, la N funcții - N aplicații

User Experience este atât de mare, încât este nevoie de o serie de artificii tehnice pentru a facilita orice fel de „reîmprospătare" a experienței utilizatorului.

În acest fel ar putea fi caracterizată în linii mari situația în care se află soluțiile *enterprise* la momentul de față: un punct de cotitură, care trebuie gestionat cu foarte multă atenție de către fiecare producător de astfel de soluții, orice pas greșit în această privință putând însemna pierderea clienților în favoarea competitorilor care au reușit să profite mai bine de noile tehnologii.

În continuare, vom realiza un studiu de caz în care vom analiza modul în care SAP, liderul global în ceea ce privește soluțiile *business enterprise*, a știut să răspundă noilor cerințe din piață, prin definirea unor strategii de adopție a tehnologiilor *state-of-the-art*.

SAP se află printre companiile care au reușit să identifice (sau chiar să preconizeze) aceste tendințe în urmă cu câțiva ani, putând astfel să pregătească din timp terenul pentru schimbările ce vor urma.

În 2013, SAP a pus la punct o strategie de User Experience [4] ce are la bază trei mari piloni:

- NEW: Dezvoltarea tuturor noilor aplicații respectând noile norme de „usability".
- RENEW: „Împrospătarea" experienței aplicațiilor existente prin interfețe noi.
- ENABLE: Sporirea posibilităților de extindere ale aplicațiilor ce stau la dispoziția clientului.

Această strategie specifică tipurile de tehnologii care trebuie folosite în funcție de context și care sunt regulile ce trebuie respectate în dezvoltarea

aplicaţiilor.

În continuare, din multitudinea de soluţii tehnice menţionate în cadrul strategiei de dezvoltare SAP vom aminti câteva, care vin să soluţioneze tocmai problemele semnalate în prima parte a acestui articol: SAP Fiori, UI5 & Netweaver Gateway.

La origine, SAP Fiori a fost numele dat unui val de noi aplicaţii concepute pentru a acoperi cele mai frecvente Use Case-uri întâlnite în practică. Specific acestor aplicaţii sunt interfeţele intuitive, toate construite după același tipar, cu ajutorul *framework*-ului JavaScript SAP UI5. Acesta la rândul său poate fi privit ca o extensie a popularului *framework* JavaScript *JQuery*, ce aduce în plus o serie de funcţionalităţi specifice SAP.

Datorită succesului acestor aplicaţii, SAP a decis să dea posibilitatea oricărui dezvoltator terţ de a implementa propriile aplicaţii Fiori. Astfel conceptul SAP Fiori a fost generalizat, așa încât în momentul de faţă acesta nu mai reprezintă „valul" iniţial de aplicaţii al SAP-ului, ci mai degrabă o serie de principii de dezvoltare comune ce trebuie respectate atunci când vine vorba de implementare de noi aplicaţii Fiori: „role-based, responsive, simple, coherent, delightful" [5]

Pe scurt, aplicaţiile Fiori oferă utilizatorului o experienţă îmbunătăţită, adaptată la nevoile acestuia (personalizată în funcţie de rolul utilizatorului în cadrul organizaţiei), prin interfeţe simple & intuitive, indiferent de dispozitivul prin intermediul căruia acestea sunt accesate.

Dar după cum am menţionat deja, metodologiile și tehnologia pe baza cărora sunt construite noile aplicaţii nu reprezintă decât jumătate din poveste. Cealaltă jumătate se referă la modul în care aplicaţiile pot fi integrate pentru a premite reutilizarea funcţionalităţilor deja existente.

Şi în cazul SAP această problemă este cu atât mai apăsătoare, cu cât în portfoliul companiei se află o mulţime de soluţii business consacrate, care însă au fost dezvoltate în urmă cu mai mulţi ani, chiar înaintea erei Web 2.0.

Aici intervine rolul componentei SAP Netweaver Gateway, concepută pentru a facilita integrarea dintre interfeţele de ultimă generaţie (v. SAP Fiori) și *backend*-ul SAP. Practic Netweaver Gateway vine cu un set de adaptere pentru diferitele tipuri de tehnologii care pot fi regăsite în *backend*-ul SAP, și permite expunerea acestor funcţionalităţi sub forma unor servicii REST care pot fi consumate de către orice client SAP sau non-SAP.

În ceea ce priveşte protocolul de comunicare folosit pentru aceste servicii, SAP a ales OData, un protocol *open*, specificat iniţial de către Microsoft [6],

ce reutilizează verbele HTTP (GET, POST..) pentru accesul la entităţile serviciilor REST. Astfel se obţine un *landscape* de sisteme în care întreaga comunicare poate avea loc pe baza unui protocol de comunicare unic, pe înţelesul tuturor entităţilor implicate.

Analizând întreaga strategie de dezvoltare a SAP, ceea ce se observă în primul rând este deschiderea către tehnologii și standarde *open*, fapt întâlnit mai rar până acum în trecutul companiei, și care sugerează că SAP a ales să mizeze și pe comunitatea de developeri non-SAP pentru dezvoltarea viitoare a produselor sale.

Ce putem însă spune fără reţineri este că în cazul SAP combinaţia acestor tehnologii s-a dovedit a fi una câștigătoare, *feedback*-ul din rândul clienţilor vizavi de noile aplicaţii Fiori fiind unul extrem de pozitiv. Pe de altă parte, privită în ansamblu, această piaţă a aplicaţiilor enterprise „consumer-grade" încă poate fi considerată una emergentă, dar cu un potenţial de creștere fabulos, care așteaptă să fie exploatat.

Bibliografie

1. 1. Wall Street Journal, „Avon to Halt Rollout of New Order Management System," - http://online.wsj.com/articles/SB10001424052702303932504579251941619018078.
2. 2. Forrester Consulting, „Key Strategies To Capture And Measure The Value Of Consumerization Of IT," - http://www.trendmicro.com/cloud-content/us/pdfs/business/white-papers/wp_forrester_measure-value-of-consumerization.pdf.
3. 3. Forbes, „Time To Reset Enterprise Software's Future And Focus On Business Value First," - http://www.forbes.com/sites/louiscolumbus/2014/03/26/time-to-reset-enterprise-softwares-future-and-focus-on-business-value-first/.
4. 4. SAP, „SAP User Experience Strategy," - http://www.sapdesignguild.org/User_Experience_Strategy_2013_10.pdf.
5. 5. SAP, „Fiori Guidelines," - http://experience.sap.com/fiori-guidelines/.
6. 6. „OData Protocol Specification," - www.odata.org

Autor

Victor Ionescu - Senior IT Consultant, SAP Development
@msg systems Romania

Simona Bonghez, Ph.D.

Cu o pregătire teoretică şi experienţă practică temeinice în Management (expertiză în Managementul schimbării, Managementul proceselor, Design Organizaţional, Management strategic şi planificare de afaceri), Simona a realizat numeroase programe de instruire şi perfecţionare în acest domeniu (inclusiv dezvoltarea materialelor suport), completate cu servicii de consultanţă (Analiza situaţiei existente, dezvoltarea afacerii, realizare plan afaceri, management al proiectelor de trasformare si repoziţionare radicală) în România, Moldova, Republica Cehă, Slovacia și Austria.

Experiența de peste 20 ani în Management de proiect (trei certificări internaționale în management de proiect oferite de Project Management Institute, International Project Management Association și UK Office for Government Commerce), dublată de experiența de peste 12 ani în instruirea adulţilor şi livrare de traininguri de management de proiect, reprezintă un avantaj personal al Simonei.

Are un doctorat în sociologie și este, de asemenea, *trainer* acreditat SDI (Strenght Deployment Inventory).

15

ESTUL ÎNTÂLNEȘTE VESTUL INTELIGENȚA CULTURALĂ ÎN PROIECTE

CULTURA ÎN CONTEXTUL MANAGEMENTULUI DE PROIECT

Globalizarea are un impact major asupra modului în care companiile își desfășoară afacerile, acest lucru fiind reflectat, de asemenea, în modul în care își gestionează proiectele. Mediul internațional îndeamnă spre o mentalitate diferită, una care înțelege și acceptă diversitatea culturală ca un fapt în cadrul proiectele noastre. Având echipe de proiect multi-disciplinare, multiculturale care își desfășoară activitatea la nivel transfrontalier, peste fusuri orare, precum și în mai multe limbi și culturi, crește și complexitatea procesului de management de proiect. Dincolo de presiunea de a realiza tripla constrângere (satisfacerea cerințelor în timpul estimat și în condițiile de cost acceptate de client) cerințele de calitate, asigurarea unei bune gestionări a echipei de proiect, asigurarea comunicării pe întreg ciclul de viață al proiectului, diminuarea riscurilor, precum și echilibrarea nevoilor și așteptărilor părților interesate (al stakeholderilor), globalizarea a adus noi provocări cu privire la cultura și diversitatea cu care managerii de proiect trebuie să se confrunte. Diferențele lingvistice pot crea confuzii în ceea ce privește sarcinile și așteptările proiectului. Deși limba engleză a devenit o limbă de circulație internațională, nu toate persoanele care lucrează într-un proiect internațional sunt la același nivel de fluență, dificultatea de a înțelege sau a traduce exact termenii și conceptele este aplicabilă atât verbal, precum și în scris. Lucrul cu parteneri din zone cu fus orar diferit creează probleme în sincronizarea membrilor echipei

și în asigurarea prezenței acestora la ședințe, unora dintre ei impunându-li-se să participe la conferințe care se desfășoară noaptea sau dimineața devreme. Există și multe alte exemple, mai ales atunci când asemenea provocări au legătură cu sponsorul proiectului sau clientul.

Una dintre cele mai mari provocări provine din diferențele culturale. Cultura este un „sistem complex de componente interdependente care trebuie să fie interpretate holistic" (Moran et al., 2014, p. 12) care acoperă valorile și credințele, influențând „conceptele oamenilor despre lumea înconjurătoare, în general despre practicile de afaceri și în special, practicile cu privire la managementul de proiect" (Stawicki, 2008). Având în vedere impactul ridicat al culturii asupra modului în care ne desfășurăm activitatea în proiecte, în special în mediul multicultural, literatura de specialitate conține modele cu ajutorul cărora diferențele culturale pot fi explicate și descrise. În acest sens, cele mai relevante sunt lucrările lui Richard E. Lewis, Edward T. Hall, Geert Hofstede, și ale proiectului GLOBE, care oferă metode complementare de a analiza cultura.

Între 1967 și 1973, Geert Hofstede a realizat unul dintre de cele mai cuprinzătoare studii despre modul în care cultura influențează valorile la locul de muncă. Analiza lui a identificat – inițial - patru grupe de valori care disting culturile țărilor unele de altele. Până la sfârșitul anului 2010, cercetarea inițială a fost extinsă în 93 de țări și alte trei grupe de valori au fost adăugate. Grupurile de valori sunt acum cunoscute ca dimensiuni ale lui Hofstede privind cultura națională: Ecart al puterii (Distanță față de putere), Individualism versus colectivism, Masculinitate versus Feminitate, Evitare a incertitudinii, Orientare pe termen lung, Pragmatism versus Normativ și Indulgență versus Reținere (www.geert-hofstede.com). Datele cercetării sunt disponibile pe *site* și pot fi făcute comparații între țări, relevându-se astfel acele dimensiuni care – având diferențe mari de scor – arată diferențe comportamentale care pot afecta dinamica echipei și ca urmare activitatea în proiect.

Am ales să exemplific printr-o comparație făcută – pe *site*-ul menționat - între România, Marea Britanie și Statele Unite ale Americii și voi explica doar una dintre dimensiuni: Individualismul exprimă măsura în care societatea încurajează relaţiile interpersonale și realizarea individuală. O valoare scăzută pentru individualism denotă o societate cu o natură colectivistă, membrii căreia sunt responsabili unul pentru celălalt, cu legături strânse între ei. Într-o societate cu un scor mare pentru această dimensiune (cum e

în cazul SUA) individul își poartă singur de grijă (și familiei apropiate), fără să aștepte sprijin (prea mult) din partea societății sau autorităților. Ceea ce, tradus în termeni de comportament în cadrul echipei spune că este foarte plauzibil - în cazul în care avem colegi din SUA sau Marea Britanie – ca aceștia să încerce să rezolve problemele fără a cere sprijin, spre deosebire de români în cazul cărora este mai probabil să apeleze la ajutorul colegilor, care îi vor sprijini simțindu-se responsabili și pentru rezultatele lor.

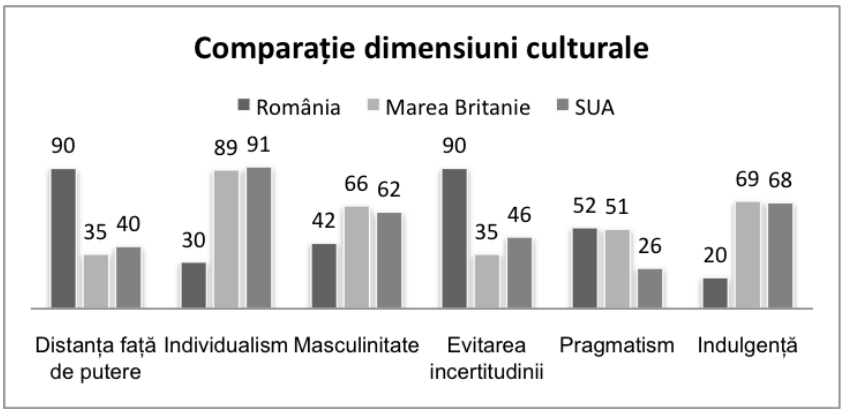


Figura1: Comparație între dimensiuni culturale (www.geert-hostede.com)

Cercetarea lui Hofstede a fost extinsă prin studiul empiric realizat de grupul GLOBE (Casa et al., 2004). Din motive conceptuale, GLOBE a transformat cele cinci dimensiuni ale lui Hofstede în nouă, validate de cei 17.000 de participanți din 62 de țări. Aceste țări au fost împărțite de către cercetători GLOBE în grupuri regionale pentru care au fost analizate asemănări și diferențe culturale, dezvoltându-se generalizări semnificative:

Un alt model ce merită să fie luat în considerare este modelul lui Lewis – Clasele culturale - care a fost dezvoltat ca o soluție practică, vizuală și eficace pentru a ajuta companiile și alte organizații internaționale să înțeleagă cauzele și consecințele diferitelor dinamici culturale din întreaga lume și modul în care acestea au un impact asupra eficienței și profitabilității afacerilor (www.riversdown.com). Richard D.Lewis a clasificat culturile lumii în trei grupe:

- Cultura Linear-Activă - Orientată către sarcini, planificatori extrem de organizați,

- Cultura Multi-Activă - Orientată spre oameni, volubili și relaționali,
- Cultura Reactivă - Introvertită, orientată spre respect.

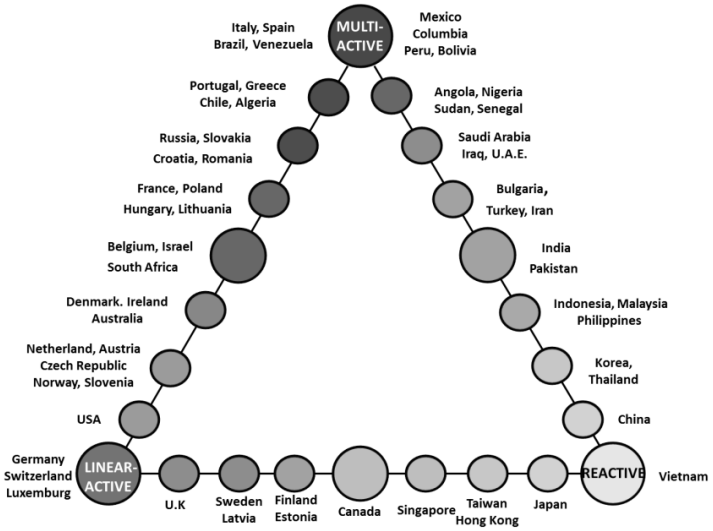


Figura 2: Modelul lui Lewis – Clasele culturale (www.riversdown.com)

Modelul Lewis include toate elementele majore ale culturii, ele putând fi folosite de către managerii de proiect sau de membrii echipei de proiect pentru a înțelege mai bine și pentru gestiona diferitele naționalități și grupuri etnice existente în echipele lor.

Într-unul dintre interviurile luate, managerul de proiect (peste 15 ani experiență în context multicultural) ne-a dat exemplul unei echipe în care membrii echipei erau români, germani, englezi iar clientul arab. Cea mai importantă dintre provocări a apărut odată cu comunicarea, dar nu numai din cauza limbii. Limba engleză este vorbită de obicei în lumea afacerilor din Orientul Mijlociu. Cu toate acestea, contextul de comunicare este cu mult diferit. Cuvintele comunicate în Occident sunt destinate pentru ceea ce sunt și răspunsurile sunt furnizate ca suport pentru ceea ce a fost spus sau scris; în lumea arabă, este opusul. Comunicarea trebuie să fie interpretată iterativ pentru a înțelege lucrurile nespuse și trebuie „citit printre rânduri". Este tipic pentru arabi să suprainterpreteze ceea ce se comunică de către omologii lor occidentali și pentru occidentali de a subinterpreta ceea ce este comunicat de către omologii lor arabi. Pentru a simplifica și mai mult, o co-

municare între arabi și occidentali s-ar termina cu ușurință cu reflecția arabă, „Ce înseamnă de fapt afirmațiile lor?", în timp ce occidentalii ar cugeta „De ce nu au răspuns la toate întrebările noastre?". Lecția învățată de managerul de proiect respectiv a fost – ca din start – să aloce mult mai mult timp comunicării, construirii unei relații, fie cu un client, fie cu membrii echipei dacă sunt dintr-o țara arabă.

Esența timpului este un alt element de luat în calcul în contextul managementului de proiect. Termenele stabilite pentru finalizarea livrabilelor pot avea sensuri diferite, de exemplu, în Occident și în lumea arabă. Managerul de proiect intervievat ne dădea exemplu un proiect în care graficul de execuție spune că „design-ul trebuie să fie finalizat până la 1 noiembrie". În Vest, termenul ar fi perceput „nu mai târziu de 01 noiembrie", în lumea arabă, însă, ar putea fi în mod obișnuit interpretat ca fiind „în jur de 01 noiembrie", în unele contexte poate însemna chiar că activitatea începe la 1 noiembrie. Un alt manager de proiect – naționalitate germană, 3 ani de experiență, în primul său proiect care a implicat membri de echipă arabi - a mărturisit că a fost surprins când a solicitat programul pentru vacanță și nu a primit date concrete, ci „o săptămână după sfârșitul Ramadanului". A fost imposibil pentru el să înțeleagă că „sfârșitul Ramadanului" nu este o dată fixă și a petrecut mult timp să caute pe internet o dată clară, specifică, până când a înțeles și a acceptat că timpul are un flux diferit pentru unii membrii din echipa lui.

Unul dintre motivele pentru care foarte multe echipe se luptă cu tensiuni între membrii săi este acela că diferențele de gândire sunt ignorate. Conștientizarea echipei privind aceste aspecte, imediat după lansarea proiectului, încurajând membrii ei să împărtășească din modul lor specific de gândire și comportament, creează un context propice colaborării.

INTELIGENȚA CULTURALĂ ȘI COMPETENȚE INTERCULTURALE. STEREOTIPURI.

Inteligența Culturală (IC) este definită ca fiind capacitatea unei persoane de a funcționa și de a gestiona situații în mod eficient în diverse setări culturale (Ang & Van Dyne, 2008, p.8). IC este similară cu IQ și coeficientul de inteligență emoțională, deoarece cuantifică un set de capabilități importante atât pentru succesul personal cât și profesional; unicitatea sa este dată de faptul că se concentrează în special pe competențele necesare pentru a fi

de succes în culturile necunoscute. Transferată în domeniul managementului de proiect, inteligența culturală este capacitatea de a manifesta competențele interculturale în cadrul echipei de proiect multi-culturale prin adaptabilitate și cunoaștere.

Proiectul GLOBE a identificat - în acest context intercultural - 22 de atribute de *leadership* apreciate în mod global, caracteristici care facilitează conducerea echipelor, precum și acele atribute care sunt văzute ca obstacole pentru un *leadership* eficient, caracteristici care împiedică efectiv conducerea eficientă a echipelor.

Valued Leadership Attributes			Obstacles to Effective Leadership
Trustworthy	Just	Encouraging	Loner
Foresight	Plans ahead	Motive arouser	Irritable
Positive	Dynamic	Dependable	Ruthless
Confidence builder	Motivational	Effective Bargainer	Asocial
Intelligent	Decisive	Informed	Nonexplicit
Win-win problem solver	Communicative	Team builder	Dictatorial
Administrative skilled	Coordinator		Noncooperative
Excellence oriented	Honest		Egocentric

Tabelul 1: Atribute pozitive și negative ale liderilor (Northouse, 2007, p.322-323)

Competențele interculturale au mai multe fațete așa cum este evidențiat în figura de mai jos. Informațiile au fost obținute în urma unor sondaje prin telefon de la persoanele responsabile pentru deciziile de angajare din companii din sectorul public, sectorul privat și ONG-uri care lucrează în context multicultural; baza: global (n = 367).

Orice persoană are o inteligență culturală specifică (IC). Acesta poate fi evaluată și îmbunătățită ca parte a învățării interculturale. Nu este ușor, nici simplu - a declarat unul dintre analiștii de business intervievat (de naționalitate română, peste 5 ani de experiență în proiecte globale) – nu este ca și cum ai decide între o stângere de mînă și o plecăciune sau a purta un văl sau nu, înseamnă să fii capabil să analizezi neînțelegerile și să le corectezi, sau să le poți evita de la început. Unul dintre managerii de proiect intervievați

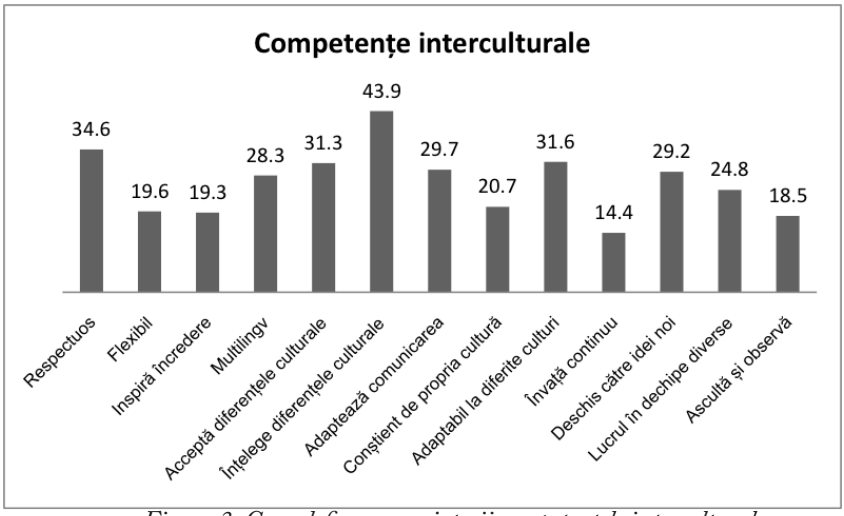


Figura 3: Cum definesc angajatorii competențele interculturale (http://infogr.am/intercultural-skills?src=web)

a folosit un cuvânt interesant când a fost întrebat despre abilitățile interculturale: „Worldmindedness". El s-a referit la o conștientizare globală a altor culturi și oameni, trăsătură care poate fi învățată. Potrivit Merryfield et al, „worldmindedness", de multe ori începe ca o conștientizare la nivel global și crește în momentul în care persoanele încep să aprecieze punctele de vedere, experiențele și viziunile asupra lumii ale celorlalți, în special cele destul de diferite de ale lor (Merryfield et al, 2008). Parafrazându-l pe Merryfield, din perspectiva managerului de proiect, fiind „worldminded" înseamnă că acesta a dobândit obiceiul de a gândi despre efectele deciziilor sale asupra tuturor *stakeholder*-ilor din proiect, având în vedere consecințele deciziilor din proiect dintr-o perspectivă mai largă. De asemenea, fiind un manager de proiect *worldminded* înseamnă că el sau ea începe să utilizeze „noi" pentru a se referi nu numai la membrii echipei, dar și la parteneri, la clienți, furnizori, utilizatori. Managerul de proiect cu peste 20 de ani experiență în proiecte complexe, a considerat implicarea în echipe multi-culturale ca fiind o oportunitate pentru educație și dezvoltare culturală. El a menționat, de asemenea, creativitatea ca fiind o abilitate necesară în găsirea de modalități acceptabile și eficiente de a motiva membrii echipei care provin din medii culturale diferite.

CONCLUZII

Globalizarea este un proces complex cu fațete multiple, iar impactul ei asupra abilităților cerute managerilor de proiect și membrilor echipelor de proiect este din ce în ce mai pregnant. Se pare că mai degrabă decât să unifice lumea, globalizarea divizează culturile, construind un mediu internațional, multicultural în aproape orice proiect, reclamînd o stare de spirit diferită, una care înțelege și acceptă diversitatea culturală ca un fapt, ca un context pentru proiectele noastre. În aceste condiții cresc – din ce în ce mai mult - cerințele privind competențele celor implicați în proiecte. Pentru a sprijini managerii de proiect în gestionarea noilor situații cu care sunt confruntați, pentru a-i ajuta să facă față diferențelor culturale în mod eficient, un prim pas este creșterea gradul de conștientizare: înțelegerea conceptului de cultură, a caracteristicilor acesteia și a efectelor ei asupra dinamicii echipei de proiect și a alegerii căilor optime de a gestiona proiectul.

REFERINȚE

1. Ang, S., Van Dyne, L. (2008), *Handbook of Cultural Intelligence: Theory, Measurement*, *and Applications*. www.culturalq.comds ,3
2. Hofstede, G. (1999), *Culture's Consequences: International Differences in Work-related Values, Sage Publications, Thousand Oaks,* CA
3. Hofstede, G, Hofstede, G.J., Minkov, M. (2010), *Cultures and Organizations: Software for the Mind. Intercultural Cooperation and Its Importance for Survival, McGraw Hill,* US.
4. Merryfield, M.M., Lo, J. T-Y., Po, S.C., Kasai, M., (2008), *Journal of Curriculum and Instructions*, January 2008, vol.2, no.1.
5. Moran, R.T, Harris, Ph.R., Moran, S.V., (2012), *Managing Cultural Differences: Global Leadership Strategies for Cross-Cultural Business Success, Eighth Edition, Elsevier Inc*, UK
6. Northouse, P. G. (2007). *Leadership: Theory and Practice.* 4th ed., Sage Publications, Thousand Oaks, CA
7. http://geert-hofstede.com/dimensions.html visited on July 24, 2014
8. http://www.tlu.ee/~sirvir/IKM/Leadership%20Dimensions/globe_project.html visited on July 24, 2014
9. http://www.riversdown.com/cross-culture/the-lewis-model/ visited on July 24, 2014
10. http://www.nafsa.org/_/file/_/theory_connections_adjustment.pdf visited on August 20, 2014

Autor

Simona Bonghez, Ph.D. – Managing Partner @ Colors in Projects

Dan Suciu, Ph. D.

Dan Suciu este lector la Facultatea de Matematică şi Informatică a Universităţii Babeş-Bolyai unde susţine atât cursuri tehnice de bază (Baze de date, Gestiunea tranzacţiilor şi baze de date distribuite) cât şi cursuri dedicate metodologiilor de dezvoltare şi coordonare a proiectelor software (Gestiunea proiectelor software, Analiza şi gestiunea sistemelor informatice complexe).

De mai bine de 5 ani de zile colaborează cu compania 3Pillar Global unde a avut ocazia să aplice în practică o mare parte din elementele ce compun cursurile sale. La 3Pillar ocupă în prezent poziţia de Director of Technical Training.

În viaţa personală este un mare consumator de filme de animaţie şi jocuri video.

16

DESPRE ANGAJAMENTE ÎN PROIECTE AGILE

Sunt conştient că titlul este din start unul care poate să "agite spiritele". Aceasta deoarece auzim de multe ori că în Agile ideea de angajament (*commitment*) este depăşită, lăsând loc altor termeni ce exprimă mai bine rezultatul colaborării continue între client şi echipa de proiect pentru implementare produsului dorit. Acest lucru nu a rămas doar la nivelul de idee. În 2011 s-au adus câteva modificări importante ghidului Scrum care se bucură la ora actuală de prestigiul de a fi una dintre cele mai populare metodologii Agile. Una dintre aceste modificări constă în înlocuirea integrală a cuvântului *commitment* cu o alternativă mai puţin dură cum este cea de predicţie sau prognoză (*forecast*).

UNII LUCREAZĂ, ALŢII PLĂTESC

Nu doar denumirea care ne este propusă spre utilizare are o nuanţă mai puţin rigidă, dar şi mentalitatea sugerată este una diferită. Nu ne mai concentrăm pe "**când**" se pot livra anumite funcţionalităţi ci pe "**ce**" funcţionalităţi se pot livra într-un interval (de obicei scurt) de timp.

Pe de altă parte însă, orice potenţial client are un plan, o perioadă de timp în care trebuie să pună în aplicare acel plan precum şi o sumă de bani maximă (buget) pe care o poate cheltui în acest scop. Cu alte cuvinte are nişte obiective foarte precise. Decizia pe care o ia în alegerea unei companii care să îi implementeze o soluţie depinde într-o foarte mare măsură de capacitatea acesteia de a se încadra în constrângeri şi de a îndeplini aceste obiective.

Ca lucrurile să fie şi mai complicate, experienţa ne arată că mulţi dintre clienţi sunt departe de a avea o idee clară cu privire la soluţia *software* ce urmează să fie implementată. Cu toate acestea, ei au nevoie de un angajament şi în absenţa acestuia vor renunţa uşor la cei care enunţă termene de livrare vagi.

Cu alte cuvinte estimările şi predicţiile se află în responsabilitatea celor care dezvoltă o soluţie *software* în timp ce obiectivele în mod natural sunt stabilite de către cei care plătesc pentru soluţia *software*. Angajamentul, în acest context, este rezultatul unei negocieri şi îl reprezintă înţelegerea sau contractul dintre cei care plătesc şi cei care realizează munca într-un proiect.

CARE E PROBLEMA CU ANGAJAMENTUL?

Există o serie de argumente pro şi contra angajamentului în Agile. Voi enumera câteva din cele care privesc angajamentul ca fiind "nociv" într-un mediu *agil*.

În primul rând se creează o relaţie de parteneriat între cele două entităţi aminitite mai sus (echipa de dezvoltare a proiectului şi clientul care plăteşte munca de dezvoltare) care din start ştiu că:

- cerinţele enunţate nu sunt neapărat complete, corecte şi finale;
- estimările care se pot face au un grad ridicat de incertitudine;
- probabilitatea ca echipa de dezvoltare să-şi schimbe în timp configuraţia este foarte ridicată.

Oricare dintre părţi ar trebui să se ferească de orice angajament în aceste condiţii, ştiind din start că probabilitatea de a fi încălcat este foarte mare. Totuşi angajamentul este necesar începerii unui parteneriat (în marea majoritate a cazurilor) şi negocierea asupra angajamentului va continua pe tot parcursul derulării proiectului.

Este evident că orice echipă își poate lua un angajament pe rezultatul propriilor estimări şi predicţii, dar probabilitatea de respectare a acestuia nu va fi cu nimic mai ridicată decât în cazul unui proiect *waterfall* clasic. Acest lucru aduce frustrări în ambele părţi făcând ca echipa de dezvoltare să fie tentată să declare că "*Agile nu e cu nimic mai bun ca* waterfall", în timp ce clientul trece rapid peste faptul că, probabil, calitatea a ceea ce a primit este mai mare şi rămâne cu senzaţia că a primit mai puţin decât ar fi trebuit.

Unul dintre lucrurile general valabile în psihologia umană este influenţa exagerată a aspectelor negative. Iar contextul specific unui proiect *software*

face ca angajamentul să fie o sursă constantă de astfel de aspecte negative.

Desigur că angajamentul nu aduce doar probleme într-un context agil. Printre altele, el responsabilizează echipa de dezvoltare şi o ajută prin inducerea unui stres pozitiv să se concentreze mai eficient pe ceea ce are de făcut.

CUM E NATURAL SĂ FIE?

Cu siguranţă aplicarea întregii metodologii clasice de gestiune a proiectelor în coordonarea proiectelor *software* a fost un proces inevitabil dar artificial. Spun că a fost inevitabil pentru că la momentul respectiv nu se întrezăreau atât de evident caracteristicile specifice proiectelor *software* ce le faceau "incompatibile" cu procesele riguroase şi rigide ale managementului clasic. Ulterior, valorile care au stat la baza apariţiei ideii de dezvoltare agilă a softului, acele valori care au reprezentat esenţa manifestului Agile din 2001, şi-au avut rădăcinile în observarea atentă a acestor incompatibilităţi.

Voi face apel la propria experienţă pentru a da câteva exemple de abordare *waterfall-like* a proiectelor *software*. De obicei lucrurile urmau cu o frustrantă frecvenţă următorul şablon: atunci când ceva nu mergea bine în proiect cineva trebuia să îşi corecteze corespunzător comportamentul astfel încât procesul să poată funcţiona. Dar comportamentul continua să se repete!

De exemplu, dacă specificaţiile erau incomplete sau neclare, clientul trebuia să se străduiască mai mult să ridice calitatea acestor specificaţii. În caz contrar, cum se aştepta la un produs de calitate, dacă cerinţele sale nu erau de calitate? Cu toate acestea, într-o etapă ulterioară a colaborării sau în următorul proiect clientul "greşea" din nou şi specificaţiile sale arătau la fel de prost.

Atunci când apăreau pe parcursul derulării unui proiect, modificări importante ale cerinţelor iniţiale, implementarea lor era amânată după terminarea proiectului conform cerinţelor iniţiale sau erau adoptate imediat (la insistenţele clientului) dar cu o reconsiderare "largă" a termenului de livrare. De fiecare dată însă scoteam în evidenţă riscurile ridicate ale unei astfel de abordări şi sugeram inhibarea oricăror alte tendinţe de a "umbla" la specificaţii. Cu toate acestea, solicitările de modificare continuau să apară.

În fine, un alt exemplu clasic era cel al estimărilor de timp necesar pentru implementarea a diverse funcţionalităţi, estimări care erau în mod repetat greşite. Chiar şi o experienţă ridicată în implementarea unor *task*-uri similare nu elimina complet erorile de estimare, obicei ce ducea la rezolvări empirice, de genul adăugării unui timp suplimentar de 10% sau 20% din estimarea

inițială, pentru ca procesul să se deruleze în bune condiţii.

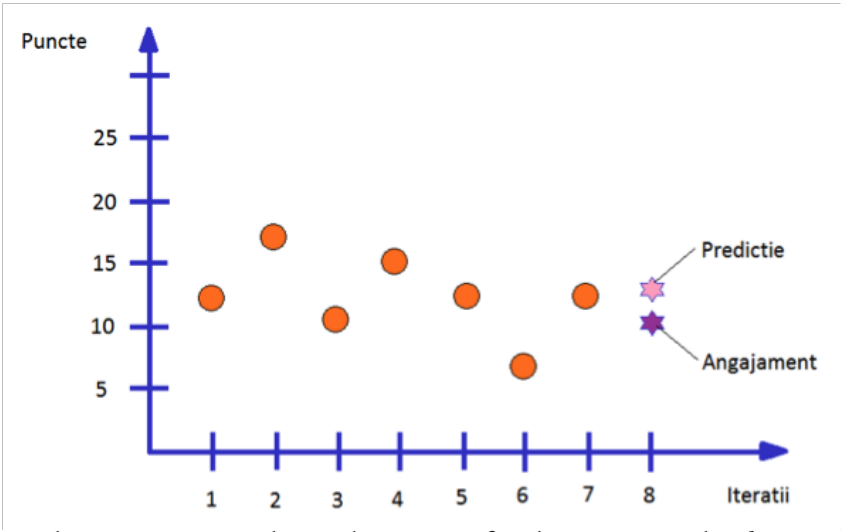
După părerea mea, unul dintre cele mai importante lucruri pe care le-a făcut manifestul Agile a fost acela de a nu considera nişte comportamente "naturale" în contextul proiectelor *software* ca fiind "greşeli". Iar consecinţa a fost aceea că principiile Agile enunţate au avut rolul de a considera aceste obiceiuri ca implicite şi au propus o schimbare de mentalitate şi de abordare, o adaptare a proceselor la situaţia dată. Astfel avem principii ca : "*Deliver working software frequently*", "*Welcome changing requirements, even late in development*", "*Working software is the primary measure of progress*." Dar luăm în considerare și procese şi practici noi ce încurajează dezvoltarea iterativă (pe principiul analizezi puţin, implementezi puţin, testezi puţin şi reiei ciclul) sau estimarea colectivă. Toate acestea transformă incapacitatea unui client de a da specificaţii clare şi complete de la prima încercare sau "apetitul" său pentru modificări de cerinţe precum şi estimarea cu o acurateţe scăzută a timpului de dezvoltare în nişte evenimente banale, de la sine înțelese şi acceptate.

Dar să revenim la tema noastră şi anume angajamentul. Consider că practica clientului de a cere echipei de dezvoltare un angajament nu trebuie nici pe departe privită ca ceva greşit, "învechit" sau ne-natural. Dimpotrivă, este perfect rezonabil ca el să ceară şi să se aştepte la un angajament. Pasul pe care ar trebui să-l facem este unul, cred eu, conform filozofiei Agile şi anume acela de a ne adapta acestei realităţi şi de a găsi procesele sau practicile potrivite pentru a anula dezvantajele angajamentului.

ANGAJAMENTUL ÎN MANIERA AGILE

Voi exemplifica modul în care putem lucra cu predicţii şi angajamente pe proiecte Agile considerând un proiect ce se află în faza de estimare a implementării *task*-urilor pentru iteraţia a 8-a. Echipa de dezvoltare estimează funcţionalităţile ce trebuie să fie dezvoltate într-o iteraţie folosind puncte de complexitate. Să presupunem că graficul de mai jos reprezintă numărul de puncte de complexitate realizate de către echipă pentru fiecare iteraţie pe durata celor șapte iteraţii care s-au derulat deja.

Considerând aceste valori şi calculând media punctelor realizate per iteraţie obţinem 12.42, putem să facem o predicţie pentru iteraţia a 8-a şi să spunem că vom acoperi 13 puncte de complexitate. Evident, constatând deviaţiile în cazul iteraţiilor precedente putem să propunem intervalul dintre 12 si 15 puncte de complexitate. Cu siguranţă însă că aceste valori nu le putem folosi ca angajament. În primul rând pentru că este vorba de un



interval, iar angajamentul va trebui să se refere la un număr clar de puncte pe care ne propunem să le acoperim. Pe de altă parte din grafic rezultă că doar în aproximativ o jumătate din iteraţii am reuşit să facem peste 13 puncte de complexitate, fapt ce face ca 13 să nu reprezinte nici pe departe un candidat bun pentru luarea unui angajament.

Pe de altă parte observăm că în 6 situaţii din 7 am reuşit să acoperim 10 puncte de complexitate. Prin urmare 10 reprezintă o valoare mult mai bună de a se constitui într-un angajament deoarece probabilitatea de nu-l îndeplini este foarte mică.

CONCLUZII

Scopul acestui articol a fost acela de a arăta că a prefera un termen ca *predicţia* în detrimentul *angajamentului* nu reprezintă o abordare în spiritul valorilor Agile şi că ambii termeni pot coexista cu succes în economia unui proiect *software*. Faptul că echipele de dezvoltare se simt mai confortabil şi sunt mai capabile să facă şi să lucreze cu predicţii nu reprezintă un argument suficient pentru a încerca să redefinim mentalitatea clientului şi să-l convingem că nu are nevoie de angajamente. Mai mult, practica ne arată că un istoric consistent de predicţii ne poate duce la îmbunătăţirea calităţii angajamentului pe care trebuie să ni-l luăm în faţa clientului.

REFERINŢE

1. Bob Galen - "*The Agile Project Manager — What's the Big Deal about Commitment*?", MATRIX blog, 2012 (http://blog.matrixresources.com/blog/agile-project-manager-%E2%80%94-what%E2%80%99s-big-deal-about-commitment)
2. Scott Sehlhorst - "Agile Estimation, Prediction, and Commitment", Tyner Blain articles, 2011 (http://tynerblain.com/blog/2011/08/09/agile-estimation/)
3. John Clifford – "Agile Estimation:Key Principles and Practices for Successful Agile Projects", Construx presentation

Autor

Dan Suciu Ph. D.
- Director Technical Training @ 3Pillar Global
- Lector @ Facultatea de matematică, Universitatea Babeș Bolyai

Monica Soare

Monica Soare are peste 15 ani de experienţă în managementul resurselor umane în companii de top din domeniul FMCG şi Farma, ultimii ani dedicându-i consultanţei. Monica are o abordare pragmatică, analitică, pasiune şi energie, precum şi o experienţă vastă în managementul unor proiecte mari şi diverse în domeniul resurselor umane.

Ariile de expertiză ale Monicăi includ: strategie de resurse umane, diagnoză organizaţională, recrutare şi employer branding, managementul talentelor, compensaţii şi beneficii, training, coaching, dezvoltarea angajaţilor, schimbare organizaţională şi consultanţă.

Monica Soare are peste 15 ani de experienţă în managementul resurselor umane în companii de top din domeniul FMCG şi Farma, ultimii ani dedicându-i consultanţei. Monica are o abordare pragmatică, analitică, pasiune şi energie, precum şi o experienţă vastă în managementul unor proiecte mari şi diverse în domeniul resurselor umane.

17

MANAGEMENTUL ECHIPELOR VIRTUALE

În 1999 gigantul NASA a lansat în spațiu un satelit orbitator, care la scurt timp după lansare a explodat. Această greșeală aducând prejudicii de peste 125 milioane $ Statelor Unite.

În urma anchetei s-a descoperit că la acest proiect lucra o echipă virtuală cu mari probleme de comunicare. Una era localizată în California și cealaltă în Colorado. În mare parte, comunicarea dintre aceste două echipe se făcea prin mail, eșuând în a realiza că echipa din Colorado lucra cu unități metrice iar cea din California cu unități imperiale.

Comitetul care a anchetat accidentul a descoperit că discuțiile dintre cele două echipe erau mult prea informale și că ambele echipe trăiau cu impresia că se înțeleg perfect.

Acest exemplu datează din 1999 pe când echipele virtuale erau la începuturi însă și astăzi, după 15 ani, ne confruntăm cu aceleași probleme, firește gravitatea deosebind de la o industrie la alta.

Apărute prima dată în domeniul IT&C, echipele virtuale au crescut de la o zi la alta, fiind eficiente din punct de vedere al costurilor pentru orice companie și în aproape orice domeniu.

Recent, Economist Intelligence Unit, aripa de cercetare a editorialului The Economist a realizat o cercetare asupra persoanelor care lucrează în echipele virtuale, oamenii care îi conduc și provocările cu care se confruntă.

Astfel, au descoperit că unul din trei manageri recunoște că echipele virtuale sunt conduse defectuos datorită faptului că managerii aplică aceleași practici de *leadership* precum în cazul echipelor clasice.

CÂT DE VIRTUALI SUNTEM DE FAPT?

Din ce în ce mai multe companii îmbrățișează ideea echipelor virtuale pentru reducerea costurilor sau pentru a crește flexibilitatea livrării servicii-

lor, astfel, apariția echipelor virtuale a fost o stare naturală care a apărut și s-a dezvoltat pe zi ce trece tot mai mult.

Evoluție sau nu, echipele virtuale sunt prezente aproape în fiecare companie. Studiul realizat de Forbes la începutul anului 2014, în rândul companiilor americane relata că în general, 30% din numărul total de angajați lucrează în echipe virtuale, în creștere față de 2010 când acest procent era de 23%.

CUM ARATĂ O ECHIPĂ VIRTUALĂ?

Înfățișarea ei, asemănătoare cu cea a unei echipe clasice, are între 5 și 20 de oameni. Marea majoritate sunt *project-based* și membrii acestora fac parte și din alte proiecte, în timp ce pentru alții este un *full-time* job și deloc surprinzător că majoritatea se află în industria IT&C.

Membrii echipelor virtuale sunt omniprezenți și provin în mare parte din Generația Y. Din fericire, IT-ul a spart orice barieră geografică și de timp, eliberând oamenii să lucreze oricând de oriunde și orice vor.

PROVOCĂRI

Echipele virtuale vin cu foarte multe provocări. În primul rând colegii nu se văd între ei, fapt ce împiedică stabilirea unei relații bazată pe încredere. Dacă nu este gestionată corect, această lipsă de încredere poate sabota toate obiectivele pe care această echipă le are.

O altă provocare majoră este comunicarea, în special atunci când în echipa virtuală intră persoane din alte țări sau culturi. Comunicarea prin *mail* sau telefon poate fi dificilă, deoarece lipsește contactul vizual sau alte elemente ale limbajului corporal.

Motivarea fiecărui membru în parte este un element greu de obținut precum și organizarea de ședințe, datorită diferențelor de fus orar.

Mai mult, soluționarea unui conflict este mult mai dificilă. Amânarea rezolvării conflictului se lasă de obicei cu părăsirea echipei de către oamenii cheie.

MOTIVELE EȘUĂRII UNEI ECHIPE VIRTUALE

Mulți manageri nu reușesc să creeze un mediu motivant de lucru la distanță. Acest minus aflându-se pe locul 2 în topul dificultăților gestionării unei echipe virtuale, după diferențele culturale. În condițiile actuale, membrii unei echipe virtuale pot fi din localități, țări sau continente diferite și provocarea de a colabora nu este doar din partea managementului ci și a

membrilor.

Capcanele mediului virtual pot fi: întârzierea *feedback*-ului, prejudecăți, interpretări greșite ale e-mailurilor, care s-ar putea soluționa ușor dacă neam afla față în față la o discuție.

Dacă faceți parte dintr-o echipă virtuală multe din aceste probleme vi se pot părea familiare.

CUM ARATĂ ECHIPA VIRTUALĂ DE SUCCES?

Echipele virtuale sunt separate de timp, cultură și atribute lingvistice. Echipele virtuale globale sunt foarte ușor de format și sunt foarte *agile* în natura lor. Ele pot ajuta companiile să scadă din timpii de răspuns în această piață hiper-competitivă, tocmai profitând de avantajul fusului orar și al echipei dispersate pe întreg globul.

Pentru ca o echipă virtuală să aibă succes, managerul trebuie să înțeleagă în primul rând că echipa virtuală este precum joaca între copii supărăcioși dintr-o grădiniță internațională.

Tocmai aici apare dificultatea managerului unei astfel de echipe.

Grădinița are foarte mare nevoie de acești copii deoarece sunt motorul de supraviețuire al businessului. Copiii cu un *background* foarte diferit riscă să nu se înțeleagă între ei și să se jignească constant fără a ajunge la consens, dar educatoarea asemenea managerului trebuie să împace pe toată lumea în vederea dezvoltării atât a grădiniței cât și a copiilor.

Obiective clar definite, un set de reguli bine delimitat, membri angajați proactiv în realizarea obiectivelor, dispuși să împărtășească informațiile esențiale sunt câteva dintre ingredientele esențiale succesului echipelor virtuale. La fel de important este ca membrii unei astfel de echipe să aibă abilități de comunicare foarte bune și să își ofere în mod constant *feedback*. Nu în ultimul rând o practică a echipelor virtuale de succes este cel puțin o întâlnire față în față a membrilor, măcar o dată pe an.

Dar pentru ca toate acestea să funcționeze, managerul echipei trebuie să creeze un cadru propice comunicării atât formale cât și informale, a *feedback-ului* și a înțelegerii diferențelor impuse de spațiu și timp.

Ca urmare, am stabilit cinci direcții importante pe care managerul de succes al unei echipe virtuale trebuie să le stăpânească și să le dezvolte:

1. COMUNICAREA

Comunicarea eficientă este esențială într-o echipă virtuală. Deschiderea, comunicarea onestă nu doar că ne ajută la înlăturarea neînțelegerilor ci ne crește eficiența.

Când vorbim de comunicare eficientă, evident vorbim de ascultare activă și de dorință de cooperare.

De asemenea, comunicarea înseamnă și *feedback* care alături de veștile proaste se oferă cel mai bine între patru ochi. Când nu este posibil, încercați să folosiți Skype, Video Chat-ul sau telefonul, în niciun caz *e-mailul*.

2. CULTURA

Diferențele culturale sunt tocmai valoarea adăugată a echipelor virtuale dar în același timp pot crea probleme.

O modalitate bună de abordare a diferențelor culturale este învățarea câtor mai multe lucruri despre *background*-ul membrilor echipei. Acest fapt implică o cercetare sârguincioasă din partea managerului care să pună în valoare aceste diferențe.

3. CONSTRUIREA RELAȚIILOR

Când faci parte dintr-o echipă virtuală trebuie să faci un efort suplimentar în ceea ce privește relațiile.

Menținerea încrederii, deschiderea, răspunsul prompt la telefoane sau *e-mailuri* este esențială. Un răspuns rapid le arată colegilor tăi că îți pasă de problemele lor. Ca manager, fii sensibil față de colegii aflați în zone cu fus orar diferit. Dacă o întâlnire este stabilită prea târziu ține minte că unii vor fi mai puțin vocali și angrenați în discuții decât ceilalți.

Folosește-te de uneltele pe care le ai la dispoziție. Folosește-te de Facebook sau LinkedIn pentru a vedea care este *background*-ul colegilor sau ce preferințe au. Apoi măcar o dată pe an oferă posibilitatea ca echipa ta să se întâlnească și față în față.

4. COOPEREAZĂ CU IZOLAREA

Profesioniștii care lucrează în echipe virtuale se confruntă de multe ori cu sentimente de singurătate și izolare, care nu sunt deloc productive. Sunt persoane care datorită faptului că nu lucrează în clădirea organizației vor simți că nu fac partea din ea sau vor crede adeseori că nu sunt importanți pentru organizație. Aceste situații sunt și mai greu de combătut dacă unele persoane

au fost implicate în procese decizionale.

Dacă sunteți managerii unei echipe virtuale, luați pași activi în combaterea izolării unor membri. Formați un grup online în care să se discute chestiuni non-formale între membrii echipei. Un loc unde își pot trimite *link*-uri sau să povestească pe scurt ce au făcut în weekend. Din punct de vedere tehnic nu mai este o provocare ca membrii echipei să se conecteze virtual. Fiți creativi și folosiți-vă instrumente care vă stau la dispoziție fie ele și rețele sociale.

5. ADAPTAREA

Secretul managerului virtual de succes stă în abilitatea lui de a previziona schimbările proiectelor sau a pieței forței de muncă virtuală și de a se adapta la schimbările rapide care se produc în industrie și în echipa lui.

În urma unui studiu aplicat asupra mediului și al tendințelor de management în echipele virtuale am ajuns la 15 de tehnici pe care fiecare manager virtual ar trebui să le aplice pentru a avea rezultate excelente în managementul echipei sale:

1. Recrutează oamenii potriviți pentru echipa ta. În cazul unei echipe virtuale fiecare persoană trebuie să fie precum o piesă de puzzle, perfect îmbinată cu celălalte. Contrabalansează lipsurile echipei virtuale cu oameni care se potrivesc și care au abilitățile de comunicare și inteligență emoțională foarte bine dezvoltate.
2. Alege în echipa ta oameni care au mai lucrat în echipe virtuale sau care au mai lucrat împreună cu succes în alte proiecte.
3. Tratează echipa virtuală la fel ca pe o echipă clasică însă acordă atenție înzecită fiecărei discuții sau e-mail. La fel ca într-o companie, membrii echipei sunt motivați să se dezvolte. O discuție periodică, între patru ochi în care să se discute planul fiecăruia de dezvoltare va fi foarte bine primită. Oamenii vor simți că managerul este sensibil față de direcția în care doresc să se dezvolte și compania vrea să investească în ei.
4. Investește într-o platformă online de comunicare interioară, asemănătoare rețelelor de socializare. De asemenea, înainte de fiecare întâlnire pregătește agenda însă, lasă timp și pentru construirea relațiilor. Datorită separării fizice, o mare provocare a acestor echipe este imposibilitatea de a învăța unii despre ceilalți lucruri esențiale și despre rolul pe care fiecare îl au în proiect.
5. Asigură-te că *task*-urile sunt semnificative pentru echipă - În mod

ideal, misiunea echipei virtuale ar trebui să rezoneze cu valorile fiecărui membru - atât ca indivizi cât și ca profesioniști care vor să își dezvolte abilitățile cu importanță vitală pentru companie. Importanța muncii cu sens și a viziunii care inspiră este foarte clară, aduce rezultate vizibile în echipele virtuale. Menționăm în acest sens două exemple foarte cunoscute: colaborarea care a produs enciclopedia *online* Wikipedia și cea a creatorilor Linux, un sistem de operare cu sursă deschisă. Nici una dintre acestea două nu este o companie însă ambele sunt realizate de un număr mare de voluntari, majoritatea fiind inspirați de devotamentul și de misiunea pe care o au.

6. Stabilește obiectivele echipei - Membrii echipei trebuie să știe și să înțeleagă de ce lucrează într-o echipă virtuală și de ce sunt importanți pentru acea echipă. Dacă ei înțeleg doar rolul lor în proiect și nu impactul pentru echipă, nu mai discutăm de o echipă virtuală.
7. Stabilește regulile - Chiar și dacă membrii echipei lucrează de acasă sau din locații diferite, ei trebuie să aibă un set acceptat de reguli. Regulile de bază includ setarea orelor în care este așteptat din partea membrilor să lucreze, pauzele de masă, determinarea ședințelor, care sunt obligatorii pentru toți membrii.
8. Obține tehnologia necesară - Echipele virtuale există de ceva vreme, cu toate astea, trendul s-a accelerat în ultimii ani. Tehnologia sprijină din plin echipele virtuale, incluzând - internetul, conferințele audio, camere video sau *folder*-e comune. Caută cea mai bună tehnologie de comunicare potrivită personalității echipei tale.
9. Caută oportunități de socializare - Echipele localizate împreună au oportunitatea de a socializa toată ziua. Echipele virtuale nu au această oportunitate, așadar managerul trebuie să se concentreze pe stabilirea de relații. Ideal ar fi ca măcar o dată pe an, toată echipa să se întâlnească pentru un *team-building*. În cazul în care nu este posibil, se poate încerca măcar pentru o perioadă, să se grupeze câte doi membri care să lucreze împreună dintr-o locație și rotativ, fiecare membru să lucreze direct cu colegii săi.
10. Politica *task*-urilor scurte - Nu delega proiecte foarte mari și lungi unei echipe virtuale. Datorită dificultăților care pot apărea în comunicare este irealist să te aștepți ca echipa să îți predea la termen proiectul așa cum ar trebui să fie. În loc să delegi *task*-uri pe 6 săptămâni încearcă să dai activități pe 2-3 săptămâni. Și atunci poți să le și spui dacă sunteți

în grafic sau nu.

11. Ții prânzuri virtuale, o dată pe lună pentru a construi un raport non formal al relației.
12. Investește în *training*-uri care să construiască comportamente de colaborare eficientă, comunicare și construirea de comunități.
13. Fii disponibil - Așa cum am amintit și mai sus, munca virtuală poate să devină izolantă. Nu lăsa membrii echipei să simtă că ești absent, fii prezent și intră în contact cu ei cât de des, nu doar în legătură cu *task*-urile de zi cu zi, ci mai mult, chestiuni legate de starea membrilor sau viața lor socială.
14. Încurajează discuțiile informale - Oamenii au tendința de a fi interesați de ceea ce fac alții în viața personală. Pentru a-i ajuta să își construiască relații puternice, încurajează echipa să își împărătească întâmplările în mod nonformal cât de des.
15. Fii creativ în sudarea echipei - De exemplu, după ce echipa ta atinge un obiectiv, organizează o ceremonie de premiere virtuală. Trimite mici cadouri membrilor echipei și pune-i pe toți să deschidă cadoul în același timp în timpul unei video conferințe.

CONCLUZII

Echipele virtuale sunt un fenomen în aflat în creștere și majoritatea executivilor au o atitudine pozitivă asupra beneficiilor acestei formule.

Persoanele care deja lucrează în echipe virtuale sunt de acord că munca virtuală le permite să colaboreze cu colegi din mai multe organizații și le oferă acces la piața globală a talentelor, lucru benefic în competitivitatea organizației și al creșterii potențialului profesional.

Un lucru cert este că modul în care trebuie gestionate nu este de ignorat, ba din contra este o provocare bine venită care ne dezvoltă autenticitatea în *leadership*.

Autor

Monica Soare - Managing Partner @ Artwin

Andreea Pârvu

Absolventă a masteratului de Psihologia Resurselor Umane, și-a început cariera ca HR Generalist la o companie de logistică, unde și-a descoperit și dezvoltat o pasiune pentru aria de Resurse Umane. Experiența internațională ca membru al echipei globale de Talent Management din cadrul unei companii de telecomunicații, a contribuit la dobândirea și îmbunătățirea cunoștințelor în acest domeniu. În plus, acest context a avut un impact pozitiv în momentul în care a decis să încerce domeniul IT ca *freelancer*. Din 2012, Andreea este Senior Recruiter la Endava - Cluj, una dintre companiile de IT recunoscută ca având una din cele mai accelerate creșteri pe piața din România.

Pe lângă activitățile corporate, a inițiat câteva proiecte de succes în domeniul educației, printre cele mai cunoscute sunt colaborările cu organizațiile studențești clujene pentru oferirea de suport studenților în definirea unui *career path.*

Totodată pentru comunitatea *mobile* din Cluj, a fost deschizator de drumuri în dezvoltarea unui proiect – Mobile Operation Systems, de îmbunătățire a abilităților și cunoștințelor tehnice. Proiectul a fost la o primă ediție, dar vor urma cu siguranță și altele. Implicarea în comunitatea de IT este dovedită și prin publicarea de articole în Today Software Magazine, una dintre cele mai cunoscute reviste de pe piața IT clujeană.

18

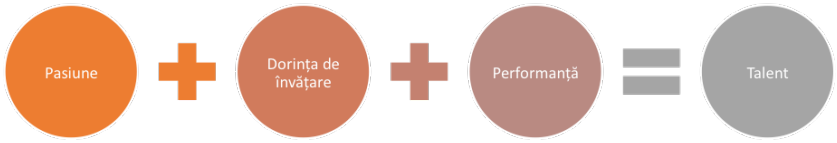
JOCUL RECRUTĂRII

Modul în care înțelegem și ne raportăm la contextul actual clujean marcat de o continuă evoluție datorită cunoscutei **dinamicități a pieței** de IT se poate constitui într-un interesant subiect de analiză. Documentarea despre dinamicitate ar fi un prim demers în seria operațiilor de analiză dedicate acestui subiect. Considerăm însă prioritară și concentrarea asupra definiției **atenției**, care este descrisă ca funcție sau mecanism de orientare, focalizare și fixare a conștiinței asupra unui obiect, sarcină, întrebare, problemă. Cele două concepte, dinamicitatea și atenția, sunt mai mult decât corelate. Pentru a înțelege cel mai bine ce se întâmplă în piața de IT este necesar să fim atenți și să ne concentrăm energia pentru a desfășura celelalte procese și structuri psihice care au un impact în activitatea noastră. Cele trei funcții esențiale ale atenției: orientarea, selectarea și concentrarea în vederea proceselor de cunoaștere, pe care le subliniază studiile de specialitate, transformate în repere importante în analiza specificului contextului IT clujean, ar putea demonstra necesitatea ***schimbării paradigmei în resurse umane.*** Modalitatea tradițională de a selecta noi angajați este cauza unor dereglări evidente în piața IT clujeană, de aceea se impune o redefinire a procesului de selectare care să valorifice teoriile moderne.

Reprezentanții de resurse umane se situează la intersecția dintre nevoia de business și nevoia angajaților, având ca rol să reprezinte ambele interese la așa numita "masă a managerilor" printr-o implicare activă în deciziile strategice ale companiei.

Schimbarea paradigmei vine din faptul că HR-ul nu mai este văzut ca funcție suport pentru organizație, ci este considerat un partener strategic. Iar pentru aceasta, primul pas care trebuie întreprins este să devină un consilier de încredere pentru a ajunge un expert în domeniul recrutării.

Încă de la început va fi clarificat conceptul de candidați activi și pasivi și care este impactul lor în tot procesul de recrutare. Dacă până în anul 2010 candidații activi erau preponderenți pe piață și aplicau la joburi, acum în industria IT situația s-a schimbat radical și numărul candidaților pasivi este într-o continuă creștere. Așadar se impune schimbarea abordării pentru că singurul aspect care a rămas nemodificat este angajarea de talente - cheia unei organizații de succes. **Talentul** este combinația perfectă între pasiune, dorința de învățare și performanța obținută.



Definiția Talentului

Schimbarea perspectivei va avea un impact major și în ceea ce privește selecția noilor angajați. În peste 70% din cazuri încă se mai întâmplă ca persoana care are cele mai bune abilități de comunicare și de prezentare la un interviu să ocupe postul. Mare parte din cei care participă la interviuri sunt candidați activi – care cred că nu necesită o descriere detaliată, ci doar o mențiune succintă, că sunt acea tipologie de angajați care caută activ pe piață. Atenția va fi concentrată asupra **candidaților pasivi.** Principala caracteristică a acestei categorii este dorința de dezvoltare a carierei prin proiecte provocatoare care să aibă un business și o arhitectură complexă, tehnologii noi și metodologii de livrare implementate corespunzător pe baza principiilor pe care le promovează. Campaniile de promovare a joburilor vacante targetează un *pool* de candidați activi și nu pasivi, pentru că oferă oportunități pe termen scurt și de cele mai multe ori poziții similare. Un *java senior developer* are aceleași specificații ale postului în orice companie. Însă un *java developer* care vede oportunități de evoluție în carieră și pași clari pe care să îi urmeze, va alege întotdeauna compania care îi oferă acest mediu.

Cum facem să atragem candidați pasivi? În cele ce urmează vor fi prezentate două soluții.

PRIMA SOLUȚIE

Nu mai folosi anunțurile tradiționale și plictisitoare!

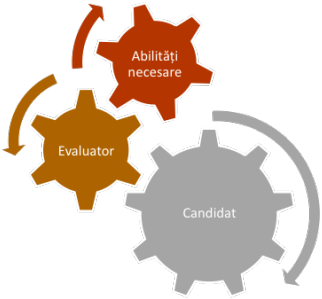
Candidații pasivi nu caută joburi bazate pe abilități, cunoștințe și experiență ci oportunități și provocări. Ca să poți face acest lucru, fiecare *job description* (fișa postului) să fie transformat într-un job cu adevărat real. *Job description*-urile sunt o înșiruire de câteva elemente: abilități, competențe, studii, iar unele cuprind și o succintă prezentare a responsilităților postului. Practic ele se rezumă mai mult la o descriere a persoanei, nu a postului. Schimbarea de paradigmă înseamnă definirea clară a posibilităților de creștere și dezvoltare, responsabilitățile și indicatorii de performanță pentru atingerea rezultatelor. Recomandarea este de a se face trecerea către niște **profile de performanță** care să fie folosite în procesul de *screening*, evaluare și recrutare pentru fiecare candidat în parte.

Joburile vacante sunt greu de găsit? Pentru că se folosesc în continuare deja mult prea cunoscutele platforme de joburi *online*. Prin schimbarea paradigmei, compania trebuie să ajungă la candidat într-un mod cât mai atractiv. În contextul în care sunt 11.000 depoziții deschise pe piață și aproximativ 9 000 de *IT professionals* candidații nu vor pierde timpul cautând joburile vacante. Compania trebuie să vină în întâmpinarea lor cu un **site atractiv** dedicat paginii de cariere – care să prezinte posibilități de creștere profesională, acces la *training*-uri, mediu de lucru prin folosirea de testimoniale ale persoanelor care deja au un rol în cadrul companiei. Mai mult decât atât **SEO** și **social media** vor juca un rol esențial în promovarea companiei pe piață. O pagină de LinkedIn sau Facebook foarte atractivă care promovează cultura și filosofia companiei, valorile și principiile va fi un atuu în poziționarea companiei ca un angajator preferat.

A DOUA SOLUȚIE

Aruncă la gunoi tot ce știi despre angajări!

Lou Adler spune că 50-60% din angajări se fac pe baza percepției pe care intervievatorul o are despre candidat. Emoții, stereotipuri, păreri personale influențează luarea deciziei. De menționat este faptul că mulți dintre candidați oferă ”piste false”, iar principalul motiv pentru care se întâmplă aceasta este că nu le sunt adresate întrebările corespunzătoare. Problema este că se acordă o atenție deosebită interacțiunii stabilite cu candidatul – ca acesta să se simtă cât mai confortabil și să nu perceapă interviul ca pe un interogatoriu și prea puțini dintre cei care recrutează se concentrează pe a afla informa-



Procesul tipic de angajare

ții cât mai detaliate despre motivație, experiență, abilități necesare, potrivire cu cultura organizațională. Nu cred că este un secret, dar majoritatea dintre aceștia iau hotărârea de angajare în primele 30 minute ale interviului. Greșit! Important este să lași informația să se sedimenteze și abia ulterior să chestionezi și să analizezi fiecare răspuns primit, până când decizia este una fermă.

Cum să faci acest lucru?

Formează-ți o părere doar la finalul interviului. Intră în fiecare interviu cu un **profil de performanță** (*performance based interview*) care să conțină elemente legate de responsabilitățile postului, de obiectivele și criteriile de performanță. La finalul interviului notează cât mai multe informații pentru a avea ancorele necesare luării deciziei.

Dacă vrei să angajezi persoane foarte bune, definește obiective foarte ridicate. Schimbarea de paradigmă înseamnă renunțarea la criteriile definite pe baza fișei postului și stabilirea unor indicatori de performanță care se așteaptă de la jobul în sine. Fișele de post sunt cele mai inflexibile instrumente care există în procesul de recrutare. În contextul dinamic al pieței, sarcinile se modifică constant pe parcursul timpului. Multe dintre fișele de post sunt învechite și mare parte din ele sunt făcute pe baza COR-ului. Într-un mediu de lucru *agile*, specificațiile se schimbă mereu în funcție de nevoia de business a clientului. Astfel, riscurile folosirii unor astfel de fișe de post pot conduce către negăsirea candidatului potrivit. Dacă nu este identificat se va ajunge în faza în care se vor face compromisuri. Un alt risc la care companiile se expun este ”clonarea” reprezentată de găsirea unei copii fidele a celor care deja există în companie.

Astfel, procesul nou de recrutare redefinit se va rezuma la trei elemente cheie.

Elementele cheie în noul proces de recrutare

Cultura organizațională a devenit în prezent o temă de interes mai ales în contextul în care asistăm la internaționalizarea fără precedent a organizațiilor. Poate fi considerată o forță invizibilă. Ca să facem o paralelă, cultura pentru organizații este ceea ce reprezintă personalitatea pentru fiecare individ. Recentele fuziuni/ achiziții (Kno – Intel, Evoline – Accenture, EBS – NTT Data) ilustrează presiunile puse pe identificarea de noi resurse, iar organizațiile trebuie să devină o prezență activă în toate domeniile.

Așadar ce este cultura organizațională?

Sathe definește cultura ca fiind *"un set de convingeri împărtășite de toți membrii"*, acestea sunt reprezentate de valorile companiei, viziunea și misiunea, simbolurile și convingerile definite și trăite.

Într-un mediu dinamic în care companiile pun din ce în ce mai mult accentul pe dezvoltarea angajaților, în ultimele decenii fiecare organizație de pe glob a încercat să definească elementele care contribuie la crearea acelui mediu organizațional de succes. Scopul lor este de a determina care sunt indicatorii de performanță specifici pentru o organizație cu un standard ridicat. Cerințele mediului extern forțează companiile să se adapteze competiției internaționale și să fie competitive atât din perspectiva prețului, calității, flexibilității cât și a relațiilor cu clienții. Managerii au dezvoltat un interes pentru crearea unei culturi organizaționale orientate spre performanță și excelență. O cultură puternică presupune alinierea obiectivelor individuale și cele de echipă cu obiectivele de dezvoltare a afacerii. Astfel învățarea va ocupa un loc cheie, iar resurse și timp vor fi alocate pentru îmbunătățirea performanței organizației.

Elementele culturii organizaționale se pot transpune în procesul de recrutare prin **trei dimensiuni.**

Dimensiuni	CE ?	CUM ?	UNDE ?
Filosofia procesului de recrutare	Definirea așteptărilor și a filosofiei de angajare	Toți evaluatorii cunosc principiile definite pe care și le însușesc	Intervievatorii sunt educați și pregătiți intern
Promovarea poziției în contextul dezvoltării organizaționale	Comunicarea către exterior a culturii organizaționale	Poziționarea companiei ca fiind prima opțiune a candidaților (employer of choice) Comunicarea strategiilor de creștere ale companiei – sustenabilitatea și posibilitatea construirii unei cariere Evidențierea realizărilor notabile ale companiei	Participarea la evenimente specifice comunității Utilizarea unor elemente vizuale care definesc cultura organizațională Folosirea de testimoniale de la angajați Crearea unui site atractiv de cariere care oferă o descriere detaliată a elementelor culturii organizaționale
Procesul de selecție- interviul față în față	Definirea indicatorilor necesari obținerii unei performanțe ridicate	Crearea așteptărilor corecte către candidați în prezentarea pașilor întregului proces	Revizuirea eforturilor depuse în procesul de selecție pentru ocuparea posturilor vacante

COGNIȚIE SAU NEUROȘTIINȚE

Neuroștiințele s-au înființat abia începând cu anii ’70. În contextul actual, neuroștiințele vor juca un rol esențial. De ce oamenii de HR trebuie să aibă cunoștințe de bază despre cum funcționează creierul uman? Cred că este mai mult o întrebare retorică. Primul motiv este legat de faptul că procesul de persuasiune este unul mult mai eficient dacă se fundamentează pe înțelegerea procesului de funcționare a creierului.

Al doilea motiv este că influențează modul în care candidații raportează propriile convingeri și credințe la cultura organizațională. Așa cum menționam și mai sus, un impact pozitiv mare îl va avea mereu alinierea culturii organizaționale cu cea a individului. Cu cât un candidat se regăsește mai mult, cu atât va fi mai atras de companie și cu atât aportul lui va fi unul mai mare. Pe de altă parte, neuroștiințele ajută la înțelegerea procesului care stă

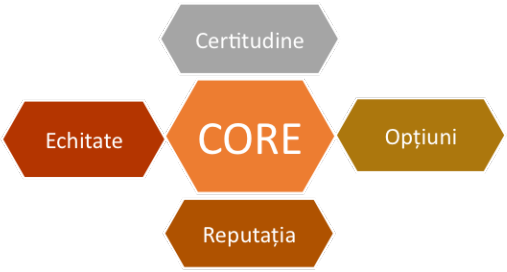
în spatele fiecăruia "De ce?" (Ex: De ce încurajăm o filosofie de recrutare? De ce un mediu în care se încurajează un angajament ridicat funcționează doar pentru anumiți candidați? De ce suntem influențați în procesul decizional?)

În cele ce urmează voi prezenta cât se poate de succint cum funcționează creierul nostru.

Structura creierului are 3 elemente importante:
Neurocortex – Creierul rațional
Creierul limbic - Creierul emoțional
Creierul reptilian - Creierul instinctiv

Ca *recruiter* conștientizăm că structura creierului nostru influențează deciziile pe care le luăm. Astfel o sugestie este ca în momentul interviurilor să dezactivăm conștient creierul limbic și cel reptilian, pentru a păstra obiectivitatea pe tot parcursul interacțiunii cu candidatul. **Rațional** ne vom concentra pe cum se vor mapa valorile și principiile candidatului în interiorul culturii organizaționale și care este valoarea adaugată a fiecărui candidat.

Am întâlnit o teorie nouă care ține de schimbarea de paradigmă în HR dezvoltată de **Jan Hills în Brain Savy HR** despre primordialitatea relațiilor și înțelegerea reacțiilor oamenilor. El propune un model care se numește **The CORE** care reprezintă patru **mari arii** care intervin în interacțiunile sociale între oameni.



The CORE – Jab Hills, Brain Savy HR

Certitudinea în procesul de recrutare poate atrage candidații cei mai buni. Informații despre planurile de viitor ale companiei, dezvoltarea unei cariere într-un mediu sustenabil sunt elemente care pot transforma compania într-o opțiune viabilă. Pentru a asigura transparența este indicată alocarea unei părți din interviu pentru a explica strategii, planuri, viziune, misiune, valori, etc. .

Oferă candidaților **opțiuni** , creează flexibilitate în abordare.

Reputația companiei pe piață poate fi câștigată prin diverse metode. Rolul *recruiter*-ilor este de a asigura transparența procesului de recrutare și prezentarea pașilor care urmează în procesul de recrutare prin oferirea de *feedback* constant și prin păstrarea legăturii cu candidații.

Ultimul element se referă la faptul că fiecare candidat își dorește să fie tratat cu **echitate** și să aibă șanse egale cu toți ceilalți implicați în proces.

COMPLEXITATE

Dinamicitatea pieței determină complexitatea procesului de recrutare. Scopul fiecărui proces de **sourcing** este de a-i identifica într-o perioada scurtă de timp cu costuri cât mai scăzute pe cei mai buni candidați de pe piață – în special pe candidații pasivi menționați în diverse paragrafe la începutul articolului. Mare parte din procesul de *sourcing* se desfășoară pe platformele *online* de recrutare. În acest fel, creativitatea este îngrădită și anunțurile arată la fel. Cum ai putea să te aștepți la rezultate performante când abordarea este una mediocră?

Schimbarea paradigmei constă în definirea unei strategii care să cuprindă mai multe surse:

Sistem integrat care să susțină procesul de recrutare și care să aibă funcționalitatea de a cumula informații din diverse surse în timp real și care să ofere posibilitatea definirii de campanii de marketing targetate diferit pe fiecare canal de *sourcing*.

Programul intern de bonusare este o strategie puternică pentru a intra în contact cu persoane performante care nu caută activ un loc de muncă. Este modalitatea cea mai eficientă de a promova cultura organizațională în mediul extern. Candidații să vină pentru că *feedback*-ul din interior este unul pozitiv și văd un impact în dezvoltarea lor personală și profesională dacă vor decide să facă parte din același mediu de lucru.

Consolidarea relațiilor cu universitățile prin încurajarea programelor de *internship* sau a celor de *training* pentru proaspeții absolvenți, prin participarea

activă în oferirea de burse și premii celor mai buni. Companiile din vest practică demult acest sistem prin care își asigură un *pool* de candidați juniori care vor fi crescuți în cultura companiei încă de la început.

Evenimente de tipul "open days" în care candidații să poată interacționa cu cultura organizației, cu reprezentanți din diferite departamente, cu mediul de lucru, proiecte, tehnologii și metodologii care pot constitui un real interes pentru cei implicați. Mai mult decât atât, astfel de evenimente încurajează activ *networking*-ul și consolidarea unei relații pe termen lung cu candidații și o mai bună cunoaștere de ambele părți a nevoilor.

Odată identificată modalitatea de promovare a poziției vacante, schimbarea paradigmei constă și în oferirea de cariere pe termen lung nu doar de joburi, așa cum menționam și la începutul articolului. Doresc să fac următoarea mențiune, criteriul financiar este și va fi mereu unul important, dar procesul decizional al candidaților are la bază **cinci factori** pe care îi văd eliminatorii:

Alinierea obiectivelor de dezvoltare personală și profesională cu cele ale companiei – persoanele performante caută provocări și oportunități care să îi ajute să evolueze.

A oferi candidatului posibilitatea de a avea o interacțiune cu șeful direct, deoarece candidații foarte buni caută să identifice lideri/ mentori care să îi impulsioneze în atingerea obiectivelor.

Amploarea proiectelor și nivelul tehnologiilor implicate, precum și echipele angrenate într-un mecanism *agile* care să ofere produse calitative clientului.

Poziționarea companiei pe piață – companiile mari și cu un *employer branding* eficient vor atrage întotdeauna candidați.

Pachetul de beneficii - care să ofere diferite facilități la transport, săli de sport, asigurări medicale și de sănătate, etc. .

Odată cu atragerea candidaților printr-un proces eficient de *sourcing*, următorul pas este modul în care este condus **interviul**. Comparația cea mai la îndemână este principiul foilor de ceapă. Fiecare strat trebuie să reprezinte un aspect care urmează să fie evaluat până când se ajunge la aspectele cele mai esențiale. În cele ce urmează va fi prezentat un decalog al **"Top 10 predictori"** folosiți în interviuri:

1. *Competențele tehnice necesare*: nu vor fi detaliate pentru că predictorul este unul mai mult decât evident în contextul în care discutăm despre industria IT.
2. *Potențialul de învățare* datorită evoluției tehnologice, tehnologii sau

framework-uri actuale, pot deveni învechite într-o perioadă scurtă de timp, iar candidatul este necesar să aibă potențialul, dar și dorința de a acumula noi informații.

3. *Motivația* – energia pe care candidatul este dispus să o investească în a dezvolta pasiune, angajament, perseverență pentru atingerea excelenței. Să nu faceți compromis niciodată la acest factor. Într-adevăr procesul de recrutare în IT este uneori greoi și mult prea provocator, dar mai bine este prelungit până când se va găsi persoana motivată să facă jobul, altfel riscurile asumate sunt mult prea mari. Insistați pe parcursul interviului pe situații în care rezultatele obținute au fost remarcabile.
4. *Inteligența emoțională* - Goleman prezintă inteligența emoțională ca fiind capacitatea unei persoane de a-și înțelege propriile emoții și ale celorlalți din grupul din care face parte. Cred că este nevoie de o schimbare de paradigmă și în ceea ce înseamnă deja binecunoscutul stereotip – că oamenii din IT nu au "people skills". Abilitatea de a fi un bun lider se aplică în două contexte diferite: unul are legătură cu capacitatea de a ghida și susține creșterea persoanelor subordonate, dar în același timp este relevantă capacitatea de influențare a deciziilor din cadrul proiectelor. Recomandarea este de a se pune accentul pe exemple în care candidatul a reușit să medieze un conflict și mai ales care au fost soluțiile găsite, dacă a reușit să schimbe o decizie importantă de business pentru că a crezut cu tărie în capacitățile tehnice.
5. *Capacitatea de a rezolva probleme* și de a identifica soluții pe baza experienței cumulate în trecut. Este esențial să poată face corelații și să anticipeze nevoile și problemele înainte ca ele să apară, iar acest lucru se poate evalua foarte ușor prin înțelegerea procesului de documentare și aflare a informațiilor relevante care pot impacta o decizie.
6. *Rezultate anterioare obținute* corelate cu obiectivele de performanță definite pentru fiecare post, așa cum am menționat chiar la începutul articolului, pentru că rezultate anterioare vor fi mereu un predictor bun pentru rezultatele viitoare. O persoană performantă în trecut ar trebui să fie performantă și în viitor. Importantă este compararea rezultatelor în medii asemănătoare, prin o bună evaluare a complexității sarcinilor de lucru, prin înțelegerea așteptărilor clienților, prin mărimea echipelor din care a făcut parte sau le-a coordonat, prin complexitatea problemelor apărute.

7. *Definirea unui trend de definire a evoluției profesionale* pe parcursul anilor și identificarea rolurilor pe care candidatul dorește să se concentreze în viitor și cum acestea se mapează pe planurile de creștere ale companiei.
8. *Abilitățile de organizare și planificare* – unul din principiile fundamentale din Scrum este capacitatea echipelor de auto-organizare, iar acest aspect este bine de evaluat pe parcursul interviului. Accentul ar trebui să fie pus pe capacitatea de a livra în intervalul de timp prestabilit, modalitatea în care se fac estimările, *user story*-uri rămase neterminate pe *sprint*.
9. *Potrivirea cu cultura organizațională* și modul în care se regăsește în viziunea, misiunea și valorile companiei.
10. *Caracterul* însumează trăsături bine împământenite care foarte greu pot fi modificate. Dacă caracterul candidatului are un impact negativ în echipa din care face parte, sugestia ar fi ca decizia de angajare să nu fie una favorabilă. Toți cei nouă predictori analizați anterior pot fi educați, pe când modificările comportamentale care țin de caracter sunt mult mai greu de schimbat, iar volumul de timp alocat este aproape dublu.

Concluzia articolului este una foarte simplă: nu e nimic mai important pentru succesul tău și al companiei decât angajarea de talente – acei "great people". Un candidat slab nu va ajunge niciodată un angajat de succes oricât de mult ai crede în acest lucru ca *recruiter*. Este important să angajezi inteligent.

Schimbarea de paradigmă a procesului de recrutare va fi elementul de succes pentru o angajare inteligentă.

Succes!

Autor

Andreea Pârvu - HR professional @ Endava

Mulțumim partenerilor noștri

AROBS Transilvania Software

AROBS Transilvania Software SA este companie specializată în dezvoltarea de soluţii software în domeniul navigaţiei GPS, aplicaţii pentru telefoane mobile şi o vastă experienţă în outsorcing pentru industria de turism, banking, transport, sănătate şi auto.

Voicu Oprean a înfiinţat AROBS în iunie 1998 cu scopul de a le demonstara tinerilor programatori că îşi pot construi o carieră de succes în dezvoltarea de software şi în România, fără a mai fi nevoiţi să emigreze. Cei 16 ani de succes şi dezvoltare pe care compania i-a cunoscut se datorează acestor specialişti IT care lucrează pentru proiecte de mare anvergură, atât pentru clienţi din străinătate cât şi pentru clienţi din România. În 2014, AROBS înseamnă o companie cu 7 sedii în ţară şi străinătate (Cluj, Tg. Mureş, Bucureşti, Iaşi, Budapesta, Chişinău, Londra) în care îşi desfăşoară activitatea peste 350 de specialişti. Aceştia susţin şi dezvoltă permanent activitatea celor 3 linii principale de business:

Dezvoltarea de soluţii software personalizate în regim outsourcing pentru companii din diverse industrii (turism, auto, sănătate, banking), inclusiv aplicaţii pentru platformele mobile Android şi iOS. AROBS este unul dintre furnizorii consacraţi de soluţii software pentru străinătate care în 16 ani de activitate a reuşit să stabilească parteneriate de lungă durată cu companii din Finlanda, Marea Britanie, Suedia, Belgia, Canada, Olanda, Germania şi SUA.

Soluţii software dedicate pieţei interne utilizate de peste 2000 de clienţi business din România: aplicaţii de monitorizare prin GPS şi management a parcului auto, aplicaţii SFA şi CRM de automatizare a vânzărilor şi de gestionare a depozitelor şi magaziilor, soluţia de automatizare a vânzării de bilete şi legitimaţii pentru companiile de transport public, primul sistem de tip TMC din România de furnizare a informaţiilor din trafic în timp real.

Distribuţie de sisteme de navigaţie GPS în România şi Europa de Est. Smailo, brandul propriu AROBS a devenit în ianuarie 2012 cel mai bine vândut sistem de navigaţie din România iar împreună cu restul brandurilor din portofoliu (Mio şi Becker) au poziţionat compania ca şi cel mai mare distribuitor de navigatoare GPS din ţară.

Succesul de piaţă din anii anteriori înregistrat cu GPS-urile Smailo a determinat compania să facă următorul pas, şi anume să crească paleta de produse prin includerea sub acelaşi brand a tabletelor Android. Cu dimensiuni ale ecranelor de 7”, 8” şi 9,7”, tabletele Smailo Web Energy sunt destinate utilizatorilor care doresc să aibă acces la Internet şi aplicaţii utile şi de divertisment, dar sunt limitaţi de buget.

„Prin tot ceea ce fac încerc să promovez un model de muncă şi viaţă bazat pe respect reciproc între oameni, clienţi şi comunitate, respect pentru viaţă şi mediu. Este un deziderat personal să creez un echilibru armonios între dedicarea profesională şi cea familială, care să se completeze cu activităţile sportive şi implicarea în comunitate.”Voicu Oprean, director general AROBS Transilvania Software SA.

Găsim cele mai bune echipe.

Le ajutăm să își rafineze ideea de start-up. Le ghidăm pe calea către succes. Le punem la dispoziție infrastructura necesară pentru a crește și înflori.

Le conectăm cu Angel Investors și surse instituționale de Venture Capital din Silicon Valley. Le conducem către măreție.

Și apoi, repetăm!

Asta facem noi. Ramâneți aproape pentru mai multe detalii în curand.

Gemini Solutions Foundry, locul în care ideile prind contur.

www.gemsfoundry.com

Mozaic Works

Think. Design. Work smart.

Mozaic Works - partenerul tău pentru creștere accelerată

Mozaic Works, înființată în 2008, este compania numărul 1 de training și consultanță pentru industriile de IT Software și Computere Science în ariile **Agile, Lean și Software Craftsmanship**, în România și Europa Centrală. Alături de companiile din Europa interesate de a dezvolta software de calitate într-un timp mai rapid, echipa Mozaic Works contribuie la creșterea rezultatelor prin **programe dedicate de training, hands-on coaching/training pe proiecte, mentoring și consultanță.**

Cu peste 500 de clienți în România, Belgia, Bulgaria, Finlanda, Franța, Germania, Suedia, UK, **Mozaic Works** este un **catalizator pentru schimbare, productivitate ridicată și învățare accelerată.**

Training și workshops (selecție): CSM (Certified Scrum Master), CSPO (Certified Scrum Product Owner), Advanced Scrum, Lean Kanban, Certified Kanban, Agile Architecture, Agile Design, Agile Management, Agile Leadership, User Stories for Agile Requirements, Test Driven Development, Unit Testing, SOLID principles, Design Patterns, Refactoring, Product Inception.

Servicii de consultanță și coaching (selecție): Agile Transformation/ Agile Adoption, Lean Kanban Transformation/ Kanban Adoption, Business agility, Seeds for innovation: Lean Startup, Product development using Agile & Lean principles, Continuous Delivery (technical practices), Agile Architecture și Incremental Design Adoption.

Clienții noștri sunt corporații naționale și internaționale, companii mici și medii, dar și startup-uri. Cu implicare în comunitățile locale din mai multe orașe din Europa, câteva fiind București, Cluj Napoca, Hamburg, Oslo, Londra, Viena, echipa Mozaic Works împărtășește cu bucurie din experiența acumulată de-a lungul timpului.

Mozaic Works website: http://www.mozaicworks.com
Mozaic Works blog: http://www.mozaicworks.com/category/blog/

Parteneri

ISBN 978-973-0-17970-5