Revista Română de Interacţiune Om-Calculator 6 (4) 2013, 323-338

Un model software cu potenţial în dezvoltarea jocurilor de strategie

Constantin Nandra, Dorian Gorgan

Departamentul Calculatoare, Universitatea Tehnică din Cluj-Napoca
Str. Memorandumului nr. 2, Cluj-Napoca
E-mail: cosmin_nandra@yahoo.com, dorian.gorgan@utcluj.ro

Rezumat. Scopul acestui articol este de a propune şi prezenta un model software care ar putea sta la baza dezvoltării jocurilor de strategie şi, în general, a aplicaţiilor interactive care implică simularea interacţiunilor dintre un set de entităţi într-un mediu dat. Modelul propus constă dintr-un pachet de componente software care definesc arhitectura şi mai multe seturi de componente interschimbabile care pot fi folosite direct în dezvoltarea potenţialelor aplicaţii grafice interactive. Acest model are potenţialul de a oferi ca alternativă la interacţiunea software bazată pe invocarea funcţionalităţii, caracteristică utilizării bibliotecilor software, un mod de interacţiune care oferă posibilitatea programatorului de a construi aplicaţia ca pe un agregat format din componente existente. În cadrul articolului vor fi scoase în evidenţă punctele forte ale acestei proiectări (flexibilitate, extensibilitate, modularizarea modelului), şi se vor prezenta şi demonstra beneficiile utilizării sale în dezvoltarea aplicaţiilor grafice interactive, cum sunt jocurile pe calculator.

Cuvinte cheie: bibliotecă software, proiectare modulară, componente reutilizabile, interacţiune alternativă.

1. Introducere

Iniţial apărute ca mijloace de recreere pentru programatori sau simple manifestări ale unor pasiuni, astăzi, jocurile video reprezintă o industrie cu venituri anuale globale imense, înregistrând cea mai semnificativă creştere din sectorul media conform (Reuters, 2011). Această creştere se datorează, în mare măsură, apariţiei microprocesorului, computerului personal şi creşterii exponenţiale a puterii de calcul din ultimele decenii, după cum menţionează Zackariasson şi Wilson (2012).

Un alt rol semnificativ în cadrul acestui fenomen de creştere a fost jucat de evoluţia metodelor de dezvoltare a acestor tipuri de software.

Este binecunoscut faptul că jocurile video reprezintă o categorie de produse software cu multă interacțiune, care tind să solicite mai multe resurse de calcul decât aplicațiile obișnuite, mai ales atunci când este vorba de simulări în timp real sau în cazul în care calitatea reprezentării grafice joacă un rol important.

Acesta este, probabil, unul dintre motivele pentru care în faza incipientă a industriei, aplicațiile de acest tip erau proiectate și implementate special pentru a folosi în mod optim resursele hardware ale platformelor precum resurse grafice sau constrângeri de memorie (Moore și Novak, 2010).

Evident, această abordare vine cu serioase dezavantaje. În primul rând, trebuie luat în considerare timpul necesar dezvoltării unei astfel de aplicații în totalitate. Un alt neajuns este reprezentat de posibilitatea destul de redusă a reutilizării codului în dezvoltarea ulterioară a unor aplicații similare.

Pentru a răspunde acestor neajunsuri, următorul pas avea să ducă la apariția unor sisteme specializate menite să îmbunătățească procesul de dezvoltare. Apărute, în general în cadrul companiilor mari, acestea încercau să ofere soluții software cât mai generale, seturi de componente reutilizabile precum și medii care să faciliteze procesul de dezvoltare. Aceste sisteme, denumite ulterior "game engines" (aprox. "motoare de jocuri"), au crescut în popularitate și număr ca urmare a introducerii elementelor de grafică 3D la începutul anilor '90 (Lily, 2009).

În zilele noastre, majoritatea sistemelor de dezvoltare oferă facilități ca: motoare de redare grafică, detecția coliziunilor, facilități pentru inteligență artificială, sunet, management la nivelul thread-urilor (fire de execuție) și al memoriei (Ward, 2008). Toate aceste elemente aduc semnificative îmbunătățiri procesului de dezvoltare a jocurilor pentru calculatoare personale și diverse alte platforme, promovând reutilizarea componentelor și oferind posibilitatea de adaptare a motorului de jocuri în scopul dezvoltării de noi aplicații interactive.

2. Obiective

Acest articol își propune să prezinte un model compus dintr-o colecție de componente software care să faciliteze dezvoltarea aplicațiilor ce implică simularea interacțiunilor dintre un set finit de entități, într-un mod asemănător motoarelor de jocuri menționate anterior.

Acest model software ar putea fi utilizat spre exemplu în dezvoltarea jocurilor de strategie întrucât acestea implică adesea simularea evoluției stării unui sistem compus dintr-o serie de entități existente în cadrul unui mediu şi controlate în mod interactiv de către un utilizator. Exemplele în acest sens pot varia de la clasicul joc de şah, până la cele mai sofisticate jocuri care permit utilizatorului să ia decizii care pot influența dezvoltarea unor civilizații virtuale.

Modelul software ce urmează a fi prezentat îşi propune să fie fundamentul unui sistem care să furnizeze două tipuri principale de facilități.

În primul rând, va oferi o soluție generală care va încerca să cuprindă funcționalitatea comună aplicațiilor menționate anterior, oferind în acelaşi timp posibilitatea de extensie.

O cerință extrem de importantă o reprezintă implementarea soluției generale într-un mod cât mai transparent cu putință. Utilizatorul (dezvoltatorul de aplicații) nu trebuie să cunoască detalii despre structura internă sau implementare pentru a putea folosi această soluție. În acest fel este redus timpul necesar procesului de dezvoltare, dar şi complexitatea soluției finale.

În al doilea rând, se urmăreşte proiectarea unei colecții de componente interschimbabile bazate pe acest model software. Această colecție va fi parte integrată a soluției ce urmează a fi furnizată utilizatorilor, şi va promova principiul de reutilizare a componentelor.

Produsul finit se adresează, pe de o parte, programatorilor, oferind în acest sens o colecție de componente software menită să uşureze considerabil efortul tipic necesar dezvoltării aplicațiilor de tipul jocurilor de strategie, iar pe de altă parte designerilor de jocuri, permițându-le acestora să realizeze prototipuri de aplicații într-un mod accesibil şi rapid.

Se urmăreşte furnizarea unui tip de funcționalitate care va permite utilizatorului, în cazul de față designerului, să specifice într-un mod interactiv structura modelului aplicației într-o formă grafică, uşor accesibilă şi intuitivă, folosind în acest sens o colecție de componente implementate şi testate ce vor fi incluse în bibliotecile livrate.

Construcția aplicației utilizând componente existente este un mod alternativ de interacțiune menit a fi folosit în conjuncție cu modul tradițional de utilizare a unei biblioteci software, caracterizat în general printr-un grad mai redus de reutilizare a componentelor software. Combinația dintre aceste două moduri de interacțiune va oferi un produs ce are potențialul de a

permite realizarea atât a prototipurilor, cât şi a aplicațiilor propriu-zise cu un minim cost de resurse şi timp.

Cu ajutorul produsului finit se vor putea dezvolta jocuri de strategie şi, în general orice simulări care implică interacțiuni între colecții de entități suportate în cadrul unui mediu. Jocurile, în general, se vor limita la partide unu la unu între doi utilizatori, sau la interacțiuni relativ simple cu entități controlate de către computer. Aceste limitări se datorează faptului că în momentul de față proiectul este în stadiul de prototip şi se urmăreşte furnizarea funcționalității de bază care este centrată în jurul unui model software destinat simulării interacțiunilor dintre entități.

Lucrarea este structurată în felul următor. Secțiunea 3 descrie conceptul motorului de jocuri, şi menționează câteva dintre cele mai cunoscute soluții existente pe piață la ora actuală. În cadrul secțiunii 4 se prezintă, în linii mari, modelul arhitectural şi funcționarea sistemului, fiind menționate câteva dintre componentele de bază şi rolurile acestora. Secțiunea 5 descrie interacțiunea dintre sistem şi utilizator, subliniind avantajele aduse de utilizarea sistemului descris în cadrul acestui articol. În secțiunea 6 se prezintă construcția unui exemplu de aplicație interactivă utilizând librăria de componente reutilizabile. Secțiunea 7 enumeră principalele concluzii ale acestui articol.

3. Studiu bibliografic

3.1. Conceptul motorului de jocuri

Dacă la începuturile industriei dezvoltarea jocurilor video implica lucrul cu resurse hardware şi software specializate, produsele fiind adesea privite ca simple jucării, în zilele noastre această activitate este la baza unei industrii foarte profitabile, ce rivalizează industria cinematografică de la Hollywood în termeni de anvergură şi popularitate.

Această dezvoltare nu ar fi fost posibilă fără apariția celebrelor motoare de jocuri „Quake" şi „Doom" de la „id Software", sau „Unreal Engine" produs de „Epic Games", care sunt, de fapt, kituri reutilizabile pentru dezvoltarea de software ce pot fi licențiate şi utilizate pentru crearea a aproape oricărui tip de joc imaginabil, după cum precizează şi Gregory (2009).

Tendința dezvoltării de astfel de kituri a prins contur odată cu apariția celebrului „Doom" la mijlocul anilor '90 (Rihal, 2007). Printe noutățile aduse de „Doom*"* la vremea respectivă se numără o grafică 3D realistă şi o arhitectură care separă funcționalitatea de bază a jocului (grafica 3D, detecția coliziunilor) de elementele specifice, cum ar fi regulile de joc, texturi, modele, etc (Lily, 2007 şi Gregory, 2009).

În zilele noastre, aceeaşi separare între funcționalitatea generală a unui joc şi conținutul său specific stă la baza dezvoltării motoarelor de jocuri moderne. Prin reutilizarea acelei părți generale a funcționalității, timpul şi costurile necesare dezvoltării de noi aplicații interactive sunt semnificativ reduse.

3.2. Soluții existente

La ora actuală se găsesc pe piață o varietate de sisteme care oferă o gamă largă de facilități pentru procesul de dezvoltare a jocurilor. De la motoare grafice şi motoare de fizică, la inteligență artificială şi biblioteci pentru comunicarea în rețea, motoarele de jocuri stau la baza dezvoltării tuturor aplicațiilor majore de acest tip.

Topul motoarelor licențiate şi utilizate pentru dezvoltarea de jocuri video de către părți terțe include nume sonore, ca „Anvil" folosit în dezvoltarea celebrelor francize „Assassin's Creed" şi „Prince of Persia"*,* „RAGE"*,* cunoscut pentru contribuția adusă cunoscutului titlu „Grand Theft Auto", sau „Unreal Engine"*,* unul dintre cele mai cunoscute şi licențiate motoare de jocuri de pe piață conform (Stead, 2009).

Datorită evoluției actuale a jocurilor video în direcția fotorealismului, majoritatea motoarelor de jocuri sunt concentrate în jurul motoarelor grafice. În general, motoarele de jocuri existente pe piață urmăresc să ofere funcționalitate orientată pe categorii de discipline.

Modelul propus în cadrul acestui articol se axează pe îmbunătățirea procesului de dezvoltare mai degrabă decât pe furnizarea de funcționalitate specializată. Astfel, spre deosebire de majoritatea soluțiilor existente pe piață, acesta pune accentul pe furnizarea unei baze de componente software care să ofere sprijin în procesul de modelare a domeniului aplicației, componente care permit interacțiuni complexe cu utilizatorul.

4. Soluția propusă

4.1. Modelul arhitectural

Modelul propus în cadrul prezentului articol este bazat pe şablonul arhitectural cunoscut ca „Model-View-Controller" (MVC, aprox. Model – Prezentare - Controlor), un tipar care promovează separarea diferitelor tipuri de funcționalități (ex: logica modelului software față de cea responsabilă cu reprezentarea grafică).

Componentele descrise în această prezentare generală sunt proiectate pentru a fi folosite pe post de şabloane. Scopul lor este de a captura acea funcționalitate care ar putea fi specifică tuturor componentelor de un anumit tip şi de a da utilizatorului posibilitatea creării propriilor componente, cu funcționalitate şi tehnici de interacțiune utilizator specifice nevoilor sale, fără a fi nevoit să cunoască detalii legate de structura internă şi implementarea componentelor de bază.

Modelul este unul extensibil şi flexibil, componente noi vor putea fi create, cu funcționalitate nouă adăugată, implementare diferită, dar compatibile cu cele vechi şi interschimbabile. Motivul pentru care specificația acestui model constă dintr-un set de şabloane interdependente este, pe lângă promovarea extensibilității şi reutilizării componentelor, modularizarea proiectului, proprietate care va permite crearea aplicațiilor folosind seturi de componente existente, livrate împreună cu biblioteca de bază, sau chiar create şi adăugate de către utilizator.

O schemă generală a proiectului propus este prezentată în Figura 1. După cum se poate observa, modelul este compus din şase elemente majore:

1. Joc – abstracție care stă la baza arhitecturii, are în general rolul de a oferi o interfață care să faciliteze accesul la model şi la prezentare către componentele din exterior. Este un agregat care are rolul de a controla accesul către componentele sale şi de a păstra consistența între stările acestora.
2. Controlor – este o componentă cu rol de comandă şi control. Comunicarea cu modelul jocului se realizează prin interfața expusă de componenta Joc. Poate consta dintr-un automat cu număr finit de stări. În general, în funcție de starea internă, de comenzile externe, şi de starea modelului, va accesa diferite părți ale funcționalității

expuse de către acesta.

3. Mediu – reprezintă componenta cu caracteristicile cele mai apropiate de Modelul din arhitectura MVC. Acesta este nucleul sistemului, iar ca structură, este un agregat constând dintr-un set de componente numite Entități care pot interacționa, pot influența Mediul sau pot fi influențate de către acesta. Desigur, starea internă a acestei componente, pe lângă caracteristicile specifice, constă din suma stărilor tuturor Entităților ce intră în alcătuirea sa.
4. Prezentare mediu – această componentă este responsabilă cu reprezentarea grafică a stării interne a Mediului. La fel ca şi componenta pe care o reprezintă, este alcătuită ca un agregat, format din componente de tip Prezentare (componente responsabile cu reprezentarea Entităților).
5. Entitate – reprezentare a obiectelor ce poate fi găzduită de către Mediu.
6. Prezentare – componentă ce are ca rol reprezentarea grafică a stării interne aparținând unei Entități. Între aceste două componente există o corespondență de unu la unu.

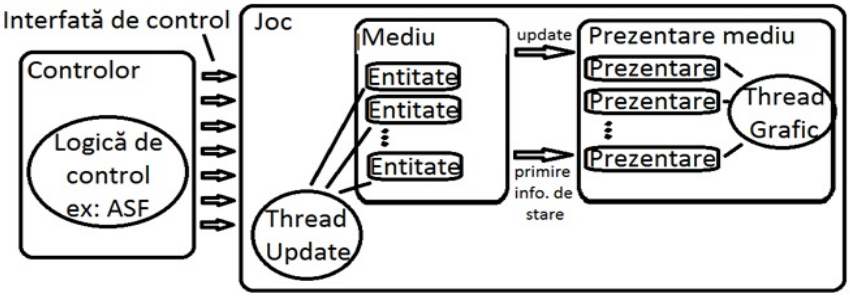

Figura 1. Prezentare generală a modelului arhitectural

4.2. Mecanisme interne

În continuare, vom examina modelul propus în detaliu, referindu-ne la subansamblul reprezentat de Mediu, Entități şi interacțiunile dintre acestea.

O Entitate este compusă din patru elemente de bază, după cum se observă în Figura 2. În primul rând, în componența ei intră o structură cu funcționalitate de coadă, folosită pe post de container pentru comenzile

primite ce urmează a fi executate. În orice moment dat, Entitatea poate avea cel mult o comandă curentă, comandă care se află în execuție. În momentul în care Entitatea primeşte comenzi noi, aceste comenzi vor fi stocate în coadă, şi vor fi livrate spre procesare (execuție) în ordinea sosirii.

Prezentarea constituie componenta responsabilă cu reprezentarea grafică a Entității. Odată ce o Entitate este creată, Prezentarea acesteia este livrată componentei „Prezentare mediu". Fiecare Entitate este responsabilă cu crearea propriei Prezentări.

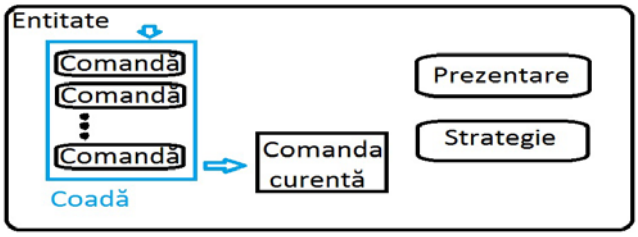

Figura 2. Sub-ansamblul componentei Entitate

Componenta numită Strategie funcționează ca o componentă de control a Entității (asemănătoare unui creier primitiv). Are rolul de a controla comportamentul acesteia în cazul în care trebuie să acționeze în absența comenzilor venite din partea utilizatorului. Strategia intră în acțiune doar în cazul în care nu există instrucțiuni implicite care să necesite execuție (coada pentru comenzi este goală şi nu există comandă curentă).

Modul de operare al Strategiei constă în trimiterea de comenzi către Entitate în funcție de parametri ca starea internă a acesteia, starea Mediului, sau a altor Entități cu care este nevoită să interacționeze. La fel ca şi în cazul Prezentării, crearea Strategiei este o sarcină a Entității.

Toate interacțiunile care au loc în cadrul Mediului, sunt realizate prin intermediul componentelor de tip Comandă. Rolul componentei este de a încapsula o serie de instrucțiuni spre a fi executate ulterior. Un alt rol, probabil la fel de important, este acela de a captura serii de instrucțiuni şi a le cataloga în funcție de scop, acestea devenind astfel accesibile şi uşor de refolosit.

5. Utilizare

Să presupunem că un programator sau proiectant de aplicații are la dispoziție un set vast de componente proiectate şi implementate după

modelul descris în cadrul acestui articol, Pentru a construi o aplicație după modelul arhitectural descris în cadrul articolului, acesta ar trebui să urmeze următoarea secvență de paşi:

- Crearea unei componente de tip Joc.
- Crearea unei componente Controlor specializată (sau alegerea uneia existente, caz în care componenta de tip Joc va implementa interfața specificată în definiția componentei Controlor aleasă).
- Alegerea unei componente existente de tip Mediu.
- Se alege componenta PrezentareMediu. În cazul în care nu este compatibilă cu componenta de tip Mediu, este creată o sub-clasă a acestei componente ce implementează interfața care specifică tipul mediului cerut în specificația componentei PrezentareMediu aleasă.
- Se aleg obiecte de tip Entitate. Pentru fiecare entitate se alege tipul de Prezentare dorit.
- Se aleg componente de tip Strategie şi Comandă disponibile pentru fiecare entitate. În cazul în care entitățile nu implementează toate interfețele cerute de comenzile şi strategia alese, pentru fiecare entitate se creează câte o sub-clasă care le va implementa.

Pentru a simplifica procesul descris anterior în termenii interacțiunii cu utilizatorul, am realizat o aplicație auxiliară care, cu ajutorul unei interfețe grafice simple şi intuitive, permite acestuia să construiască vizual o reprezentare a configurației unui joc. Această aplicație reduce interacțiunea dintre sistem şi utilizator la o simplă manipulare a unui set de componente abstracte, fără a cere utilizatorului prea multă experiență de programare sau stăpânirea conceptelor de programare orientată pe obiecte (POO).

Configurația constă dintr-un set de componente care vor fi utilizate în construcția jocului respectiv (componente reutilizabile şi componente noi, specificate de către utilizator). Componentele sunt organizate ierarhic, sub forma unui arbore, după cum se poate observa în Figura 3. Se oferă astfel utilizatorului o imagine cât mai simplă a setului de componente pe care intenționează să le integreze.

Pe baza acestei configurații, aplicația (realizată cu ajutorul tehnologiei Java) oferă utilizatorului posibilitatea de a crea un proiect de tip „Eclipse” care să cuprindă toate componentele (clasele) specificate în cadrul configurației.

Pe baza secvenței de paşi descrise anterior, utilizând această aplicație se pot genera schelete de aplicații (ex: jocuri de strategie) formate din componente existente. Deşi aceste aplicațiile nu vor fi complete, o mare parte din implementare va fi furnizată prin intermediul mecanismului de moştenire sau prin simpla invocare a componentelor existente.

Procesul de implementare va fi cu mult simplificat, întrucât dezvoltatorilor de aplicații le va reveni relativ simpla sarcină de a furniza implementări pentru metodele schelet generate automat în procesul de rezolvare a incompatibilităților dintre componente.

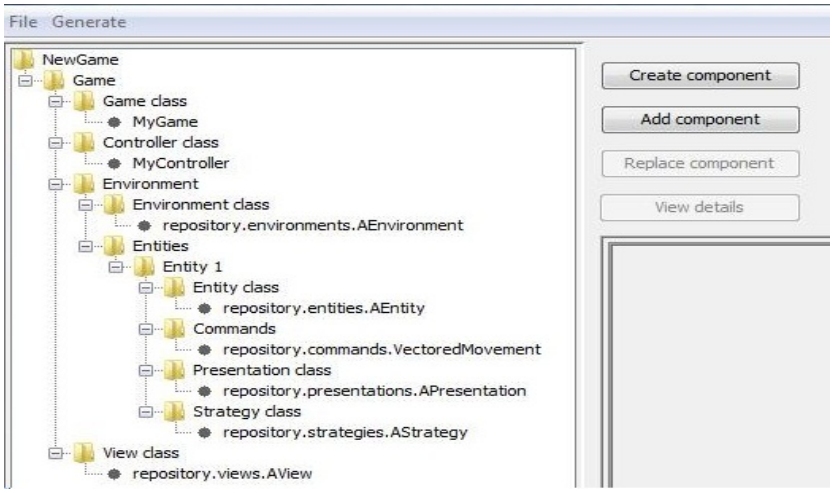

Figura 3. Exemplu de configurație a structurii unui joc realizată cu ajutorul aplicației auxiliare

Trebuie precizat faptul că aplicația auxiliară va rezolva automat aceste incompatibilități în momentul în care utilizatorul va opta să genereze un proiect nou utilizând o configurație dată. Procesul de rezolvare a incompatibilităților constă din extinderea (sub-clasarea) componentelor incompatibile şi adăugarea interfețelor necesare în definiția acestora. De aici vor rezulta metodele schelet în cadrul cărora utilizatorul va furniza funcționalitatea dorită.

Proiectul propriu-zis va consta doar din componentele create de către utilizator (Joc, Controlor etc.) şi din clasele create implicit pentru adaptarea componentelor cu interfețe incompatibile. Componentele reutilizabile vor fi făcute disponibile prin includerea bibliotecilor corespunzătoare în contextul noului proiect.

Acest mod alternativ de interacțiune dintre dezvoltatorul de aplicații şi biblioteca software are potențialul de a reduce costul, timpul şi complexitatea dezvoltării aplicațiilor.

De asemenea, s-ar putea dovedi un mod de interacțiune potrivit pentru designeri, care lucrează la nivel conceptual, fără a avea de-a face cu implementarea aplicațiilor. O astfel de funcționalitate le-ar permite să genereze proiecte schelet care să integreze componente reutilizabile. Odată generate, aceste aplicații schelet se pot finaliza într-un timp considerabil mai scurt decât în cazul aplicațiilor obişnuite.

6. Construcția unei aplicații-agregat

În continuare, vom prezenta construcția modelului unei simple aplicații utilizând componente cu caracter general, ce vor fi livrate împreună cu modelul software propriu-zis. Este vorba despre o reprezentare virtuală a unei table de şah care permite vizualizarea mutărilor posibile şi garantează respectarea regulilor ce guvernează mutarea pieselor.

Exemplul este relativ simplu, fiind prezentat cu scopul de a ilustra modul în care se pot dezvolta aplicații utilizând biblioteca software prezentată în cadrul articolului. Se urmăreşte exemplificarea paşilor enumerați anterior ce descriu modul de interacțiune alternativ în procesul de creare a aplicațiilor-agregat formate din componente reutilizabile.

Primul pas din această secvență de interacțiune presupune definirea unei clase de tip Joc (sub-clasă a clasei abstracte cu acelaşi nume). Fiind vorba despre un mod de interacțiune destinat în principal designerilor de jocuri, acest aspect (crearea unei clase) este ascuns utilizatorului. În schimb, se va utiliza interfața grafică furnizată de aplicația auxiliară descrisă anterior pentru a reprezenta structura conceptuală a aplicației şi operațiile utilizatorului (selecția componentelor existente sau crearea componentelor noi).

În acest sens, se va folosi o structură arborescentă, asemănătoare celei din Figura 4 pentru a ilustra asocierile dintre componente. După cum se poate observa în figură, configurația unei aplicații este reprezentată printr-o asociere logică de componente de diferite tipuri.

Astfel, la cel mai înalt nivel avem prezentă componenta numită Joc. Aceasta va fi unică pentru fiecare aplicație, reprezentând elementul rădăcină. Fiecare joc (aplicație) va avea în componență elemente de tip

Controlor, Mediu şi PrezentareMediu. În esență, fiecare componentă de tip Joc va fi definită de combinația de componente aleasă pentru elementele sale.

Elementele de tip Controlor sunt folosite pentru a face legătura dintre funcționalitatea modelului software şi intrarea sistemului. În termeni simplii, aceste elemente apelează funcționalitatea modelului ca răspuns la comenzile externe ca apăsarea unor anumite taste, a unor butoane ale mouse-ului, etc.

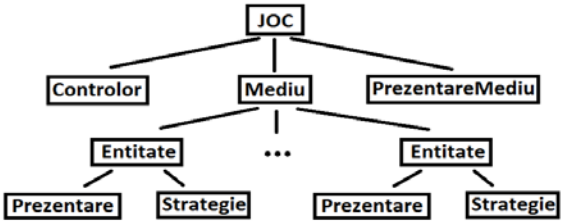

Figura 4. Reprezentarea conceptuală a unui joc bazat pe modelul arhitectural propus în cadrul articolului

PrezentareMediu este o componentă folosită pentru a reprezenta vizual starea unui element de tip Mediu ales.

Mediul, ca şi element este definit prin colecția de entități pe care o poate găzdui. Utilizatorul are libertatea de a alege elementele de tip Entitate care vor putea fi găzduite, sistemul urmând să efectueze orice modificări vor fi necesare pentru a asigura compatibilitatea dintre elementele alese.

Odată rădăcina fixată (specificarea numelui jocului), se va începe adăugarea elementelor de pe ramuri. Concret, trebuie selectate componente din categoriile Mediu, PrezentareMediu şi Controlor. În cazul de față, la fel ca în majoritatea cazurilor, se dorește crearea unei componente Controlor special concepută pentru noul joc. Asta nu oprește însă utilizatorul din a alege o componentă deja implementată în cadrul pachetului de componente software reutilizabile.

După crearea componentei de tip Joc, pe care o vom numi JocDeŞah şi a celei de tip Controlor (ControlorŞah), utilizatorul trebuie să aleagă componentele de tip Mediu, respectiv PrezentareMediu. Desigur, crearea unor componente noi în locul alegerii unora existente este întotdeauna o opțiune oferită utilizatorului.

În cazul nostru vom utiliza componenta MediuMatrice, un tip de Mediu cu dimensiuni reduse care oferă, printre altele, acces la starea fiecărei poziții

(ocupată sau nu de entități).

În ceea ce privește prezentarea mediului, vom alege o componentă numită PrezentareŞablon care oferă utilizatorului posibilitatea de a defini scheme (șabloane) de culori. Aceste șabloane vor fi extinse și asociate pozițiilor componentei de tip Mediu care urmează a fi reprezentată grafic.

Coborând un nivel, vom popula configurația mediului cu tipuri de entități ce pot fi susținute în cadrul acestuia. În acest scop vom crea un tip de Entitate nou, numit PiesăŞah. Specificația acestui tip nou de entitate va fi definită de setul de componente pe care le va susține. Vom adăuga acestei noi componente funcționalitate pentru determinarea pozițiilor permise pe tabla de șah din perspectiva poziției ocupate. Această funcționalitate va determina crearea de sub-tipuri ale componentei cu diferite implementări, care să corespundă diferitelor categorii de piese de șah (ex: regină, pion etc.)

O Entitate trebuie asociată obligatoriu cu o componentă Prezentare și un set de componente Comandă. Opțional, i se poate asocia și o Strategie.

În cazul de față, noua Entitate va folosi componenta PrezentareSimplă furnizată în pachetul de componente reutilizabile. Această componentă oferă posibilitatea de a afișa imagini simple asociate entităților pe pozițiile pe care acestea le ocupă în cadrul mediului.

Comenzile pe care PiesăŞah le poate executa vor fi reprezentate de două componente relativ simple, cu execuție instantanee: SchimbarePoziție și ÎnlocuireEntitate, ambele preluate din pachetul de componente reutilizabile.

În Figura 5 este prezentată configurația aplicației la nivel conceptual (al componentelor asociate), fiind ilustrate principalele componente reutilizabile integrate și proveniența acestora.

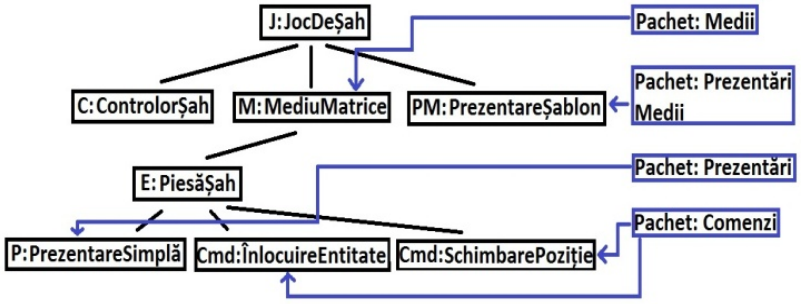

Figura 5. Reprezentarea conceptuală a modelului noii aplicații

În cadrul figurii se poate observa faptul că cinci din cele opt componente de bază ale modelului aplicației sunt componente reutilizabile. Acestea

aparțin unor colecții (pachete) care fac parte din biblioteca software furnizată utilizatorului, permițându-i acestuia să le integreze în aplicațiile dezvoltate. Procentajul componentelor reutilizabile din structura unei aplicații create în acest mod poate varia în funcție de complexitatea proiectului şi de nevoile utilizatorului.

Ideea de bază în utilizarea modelului software propus este de a folosi componente reutilizabile (furnizate în cadrul bibliotecii software) pentru cât mai multe dintre elementele structurii arborescente din Figura 4, excepție făcând, desigur, elementul rădăcină.

Utilizând o serie de componente cu funcționalitate generală livrate împreună cu modelul software propriu-zis, se poate realiza o specificație formală a structurii unei noi aplicații. Cea mai mare parte a modelului noii aplicații este construită pe baza acestei specificații.

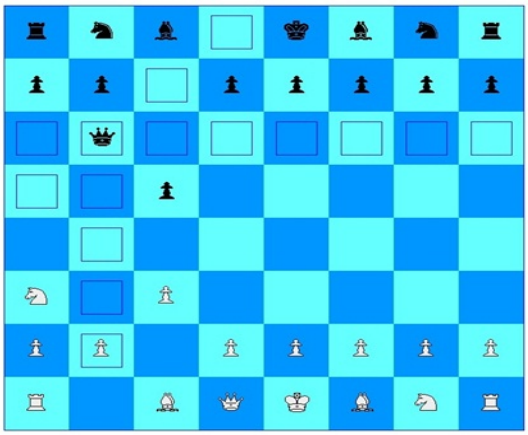

Figura 6. Reprezentare grafică a prototipului de aplicație-agregat.

Se pot astfel explora posibilele beneficii ale unui mod diferit de interacțiune dintre dezvoltatorul de aplicații şi sistem. Sistemul este reprezentat în acest caz de biblioteca software şi facilitățile oferite pentru dezvoltarea de noi aplicații. Acest mod de interacțiune oferă posibilitatea de a dezvolta vizual, la un nivel abstract (prin manipularea componentelor reutilizabile) prototipuri de aplicații, şi ar putea reduce semnificativ timpul şi efortul necesar în procesul dezvoltării acestora.

În cadrul Figurii 6 se poate observa rezultatul final, reprezentarea grafică a modelului aplicației prototip construite anterior. Aceasta permite simularea unei table de şah care încorporează regulile mutării pieselor. În

figură, celulele marcate reprezintă locațiile posibilelor mutări ale piesei selectate (regină). După cum am precizat anterior, aplicația este una relativ simplă, scopul prezentării acesteia fiind acela de a ilustra modul în care biblioteca software poate fi folosită în scopul dezvoltării de aplicații, posibilitatea de reutilizare a componentelor fiind punctul forte al acestei biblioteci software.

7. Concluzii

În cadrul articolului de față am prezentat un model software ce oferă o serie de facilități pentru dezvoltarea aplicațiilor care încearcă să simuleze interacțiunile dintre o populație de entități aflate în interiorul unui mediu dat, precum prezentarea vizuală şi interacțiunea cu utilizatorul. Mai multe detalii legate de arhitectura, funcționarea şi implementarea acestui model software pot fi găsite în cuprinsul articolului „Framework pentru dezvoltarea jocurilor de strategie" prezentat în cadrul celei de-a 10-a Conferințe Naționale de Interacțiune Om-Calculator (RoCHI2013).

S-a demonstrat faptul că modelul software prezentat este unul robust, flexibil şi modular. Acesta furnizează o soluție generală, oferind utilizatorului, în acest caz programatorului, posibilitatea de adaptare a modelului la nevoile personale. Din acest punct de vedere, modelul este unul flexibil şi promovează conceptul de reutilizare a componentelor software. Mai mult, se permite extinderea funcționalității existente, fapt ce reduce timpul necesar dezvoltării de noi componente.

Pe lângă faptul că a fost proiectat şi implementat pentru a conferi programatorilor un mecanism care să le permită dezvoltarea aplicațiilor bazate pe un model arhitectural prestabilit într-un mod complet transparent (funcționalitate tipică unui framework), acest model merge un pas mai departe, încercând să ofere designerilor şi programatorilor deopotrivă posibilitatea de a crea aplicații folosind exclusiv componente existente, livrate împreună cu bibliotecile software de bază.

Printre punctele slabe ale acestui model se numără, cel mai probabil, limitările sale în ceea ce priveşte gama de aplicații în dezvoltarea cărora poate fi folosit. Modelul poate fi folosit exclusiv în dezvoltarea jocurilor de strategie şi a aplicațiilor ce presupun simularea interacțiunilor dintre un set de entități în cadrul unui mediu dat. De asemenea, pentru a maximiza posibilitatea reutilizării componentelor software, a fost adoptat un cadru

oarecum rigid care constrânge utilizatorul să adopte un model arhitectural prestabilit, oferindu-i-se libertatea de a alege începând de la nivelul componentelor pe care le va utiliza. Din punct de vedere al timpului şi efortului necesar dezvoltării aplicațiilor, această decizie aduce avantaje clare, însă pentru utilizatorii care doresc un grad ridicat al libertății de decizie în implementarea soluțiilor se poate dovedi frustrantă.

Pentru a putea exploata pe deplin natura reutilizabilă a modelului, şi mai ales a setului de componente interschimbabile ce pot fi susținute de către acesta, a fost dezvoltat un program utilitar care permite un mod alternativ de dezvoltare a aplicațiilor. Acest program are potențialul de a reduce o porțiune semnificativă din efortul depus în cadrul procesului de creare a aplicațiilor la o simplă manipulare a unor seturi de componente reutilizabile.

Cu ajutorul unei interfețe intuitive, un utilizator fără cunoştințe aprofundate de programare (designer de jocuri) poate dezvolta un prototip al unei aplicații, la nivel conceptual, selectând componentele dorite dintr-un set furnizat împreună cu biblioteca software de bază. Acest tip de interacțiune poate fi utilizat pentru a reduce semnificativ efortul de dezvoltare al aplicațiilor interactive în termeni de resurse, cost şi timp alocat.

Referințe

Gregory, J. Game Engine Architecture. Wellesley, Massachusetts : A K Peters, Ltd., 2009.

Lily, P. Doom to Dunia: A Visual History of 3D Game Engines. Maximumpc, 2009. http://www.maximumpc.com/article/features/3d_game_engines.

Moore, M., Novak, J. Game Industry Career Guide. Delmar: Cengage Learning, 2010.

Nandra, C., Gorgan, D. Framework pentru dezvoltarea jocurior de strategie, Revista Română de Interacțiune Om-Calculator, prezentat în cadrul conferinței RoCHI2013.

Reuters. Factbox: A look at the $65 billion video games industry. uk.reuters.com, 2011. http://uk.reuters.com/article/2011/06/06/us-videogames-factboxidUKTRE7552I20110606.

Rihal, D. The History of First-Person Shooters. msu.edu, 2007.

Stead, C. The 10 Best Game Engines of This Generation. ign.com, 2009. http://www.ign.com/articles/2009/07/15/the-10-best-game-engines-of-this-generation.

Ward, J. What is a Game Engine? GameCareerGuide.com, 2008.

Zackariasson, P. & Wilson, T. The Video Game Industry: Formation, Present State, and Future. New York , 2012.