Ministerul Educaţiei al Republicii Modova

Universitatea de Stat din Moldova

Facultatea de Fizica si Inginerie

Catedra Fiziсă Aplicată şi Informatică

Lucrare de laborator Nr. 3

Tema: **“ *Diagrama de stari*”**

Lucrarea a fost îndeplinită de studentul grupei 3.2:

Timoftii Sveatoslav

Lucrarea a fost controlată de

lector univ: Diaconu Elena

Chisinau 2016

**Note teoretice:**

Toate sistemele au structura statică, descrisă de diagramele de clasă și un comportament dinamic pe care îl putem descrie prin diagrame de stare.

Diagramele de stare descriu stările pe care un obiect le poate avea pe durata vieții sale și comportamentele obiectului aflat în respectivele stări, împreună cu evenimentele care au cauzat schimbarea stării curente.

Diagramele de stare sunt folosite pentru a ajuta dezvoltatorii să înțeleagă funcționalitățile complexe ale anumitor porțiuni din sistemul modelat, arătând toate stările posibile ale unui obiect și tranzițiile care cauzează trecerile de la o stare la alta.

O diagramă de stare arată comportamentul dinamic al claselor, ca răspuns al unor stimuli externi. Nu fiecare clasă va trebui să fie modelată printr- o diagramă de stare, ci doar cele care prezintă mai multe stări potențiale pe parcursul activității unui sistem.

Principalele elemente ale unei diagrame de stare sunt:

1. **STĂRILE-** reprezintă o situație din timpul existenței unui obiect într- un anumit moment. O stare modelează o situație în care atât timp cât e satisfăcută o condiție, un anumit obiect se află într- o anumită stare care poate fi statică sau dinamică.

*Diagrama de stare pentru un obiect al unei clase* ***Factura*.**

Proprietățile stării:

1. *Name*
2. *Stereotip*
3. *Entry- action*
4. *Exit- action*
5. *Do- activity*
6. *Incomming –* listează tranzițiile ce duc obiectul la starea respectivă
7. *Outgoing –* listează tranzițiile care scot obiectul din starea respectivă
8. *Internal-* *transition* – listează tranzițiile care nu scot obiectul din starea respectivă.
9. *Composit*- *state* – stări compuse. E o stare care poate conține mai multe stări. Tranzițiile care vin sau pleacă vor fi conectate direct de una din substări. Aceste substări sunt într- o relație de tip OR. Deci, dacă un obiect se află într- un *composit- state*, atunci el va fi în una dintre aceste substări.
10. *Tranziții*- sunt săgețile care reprezintă calea parcursă de un obiect pentru a trece

dintr- o stare în alta. De obicei, o tranziție are atașat un eveniment, se reprezintă drept o săgeată orientată de la starea- sursă la starea- destinație.

1. **EVENTS-** un eveniment nu este local, deci, nu este văzut doar de o singură clasă și e reprezentat prin numele său. Un eveniment poate fi de mai multe feluri:
2. *signalEvent*- asociat cu un semnal, este produs de respectivul semnal. Semnalul, de regulă, este transmis cu ajutorul unei operații adresate unui alt obiect.
3. *callEvent***-**este produs de un apel al operației unei clase. Efectul ar fi că pașii acestei operații vor fi executați.
4. *timeEvent-*un eveniment cauzat de expirarea timpului. Timpul se calculează începând de la producerea ultimului eveniment.
5. *changeEvent*- eveniment cauzat de o expresie particulară a atributelor, care devine adevărată. O eroare poate fi considerată un eveniment, dar în UML nu există un element special de modelare a unei erori, deci, ea va fi considerată o componentă căreia i se atașează stereotipul << error >>
6. **guard-** este o condiție care este evaluată ori de câte ori are loc evenimentul. Există și posibilitatea ca o tranziție să nu aibă atașată nici un trigger, și nici un guard.
7. **Nume\_ tranziție : trigger- action [ guard ] / expression\_of\_action^send.clauses**

Un exemplu avem în diagrama următoare, unde tranzițiile de la o stare la alta au loc automat la terminarea activității interne a stării.

Evenimentele respective pot fi atașate tranziției sau nu. Tranziția are loc dacă condiția guard devine adevărată. O condiție de tip guard este evaluată o singură dată la momentul producerii evenimentului, dacă condiția e falsă, rezultă că evenimentul e pierdut.

Dacă nu există eveniment atașat tranziției, ci numai un guard, tranziția se va face când expresia booleană a guard- ului va deveni adevărată. O acțiune este o abstracție a unei proceduri care poate schimba starea unui model.

Există mai multe tipuri de acțiuni:

1. *CreateAction*- se creează o instanță a unei clase, interfețe, use- case.
2. *CallAction* – acțiune de apelare a unei anumite operații.
3. *AssignmentAction*- acțiune în care se atribuie o valoare unui atribut sau pointer.
4. *ReturnAction*- acțiune prin care se întoarce o valoare apelantului anterior.
5. *SendAction*- asociată unui semnal, cauzează trimiterea semnalului.
6. *TerminateAction*- acțiune prin care obiectul care invocă se va autodistruge.
7. *UninterpretedAction*- acțiune folosită pentru a specifica o acțiune specifică unui limbaj de programare care nu se poate clasifica în alt tip de acțiune.
8. *DestroyAction*- acțiune care distruge obiectul- țintă.

O acțiune e reprezentată într- o diagramă prin textul care o descrie. Proprietățile sunt numele, expresia acțiunii și limbajul de programare în care a fost scrisă expresia acțiunii.

1. **Pseudostates –** starea inițială este starea cînd obiectul nu are nici o variabilă care să- l descrie și nu realizează nici o activitate.

*Branch-* pseudostare folosită pentru introducerea de alternative specificate prin diferite condiții.

*Forks-* divide secvența de tranziții în subsecvențe concurente, care se execută în același timp.

*Join-* unește anumite subsecvențe.

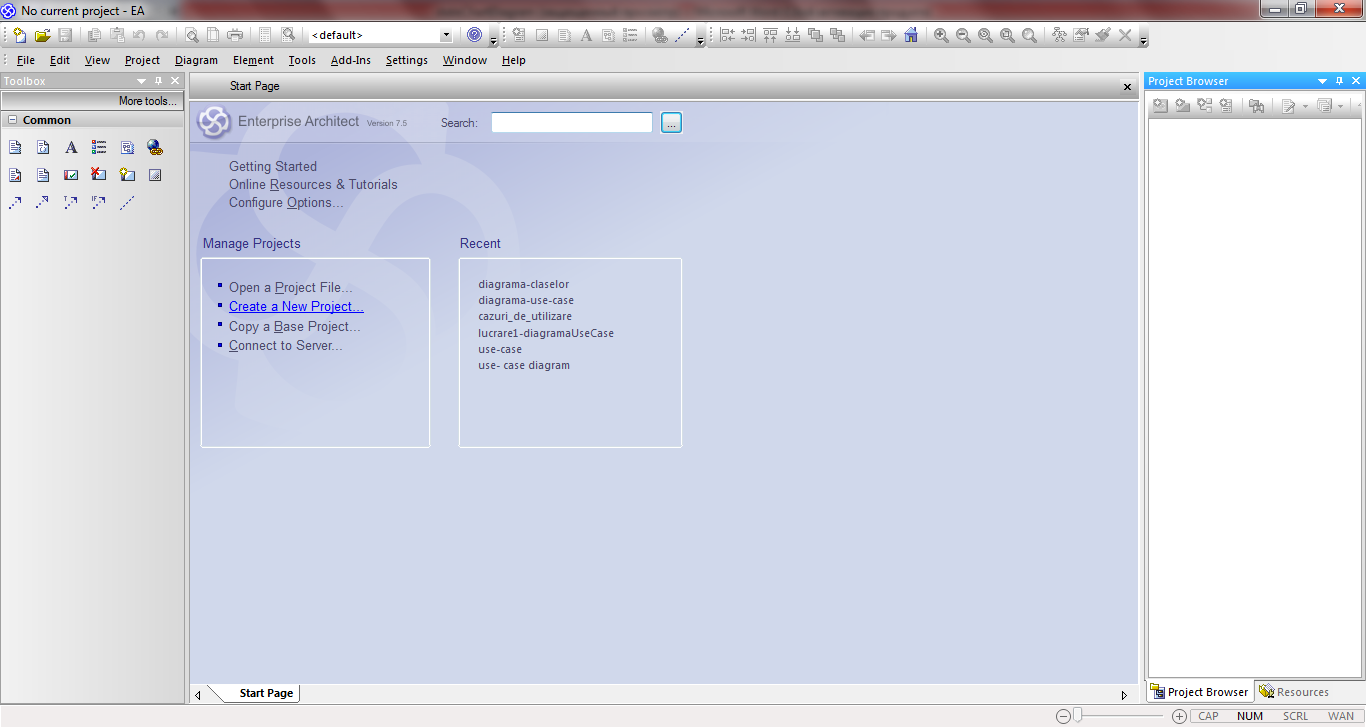
**6)History states-** Câteodată avem nevoie ca un obiect să intre într- o stare de așteptare, iar apoi, în momentul când un anumit eveniment are loc, obiectul respectiv să reintre în starea dinainte de așteptare.

**7)Starea finală-** evenimentul care finalizează diagrama de stare.

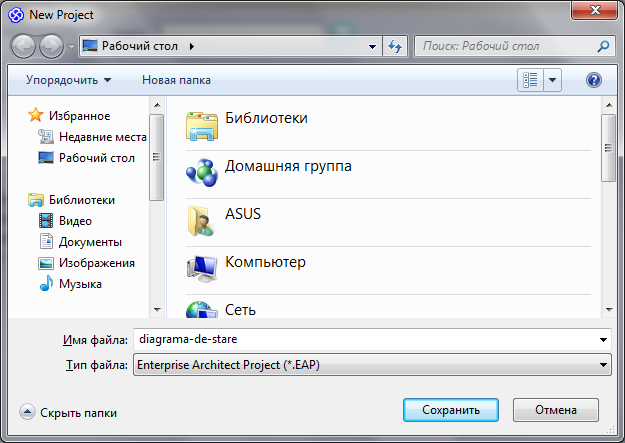
**PARTEA PRACTICĂ**

*Să se realizeze diagrama de stare pentru sistemul reprezentat de un parc de mașini.*

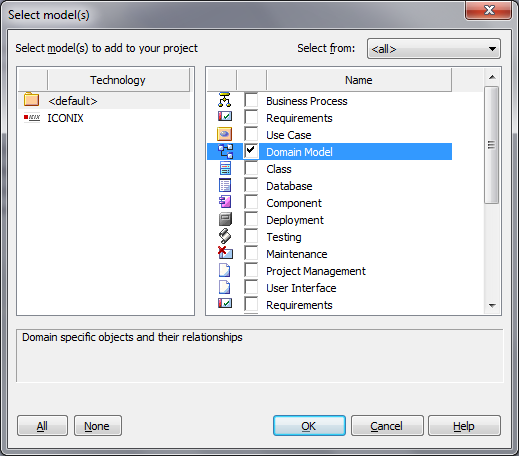
1. Deschidem programul Enterprise Architect și selectăm din pagina de start *Start Page* opțiunea *Create a New Project*



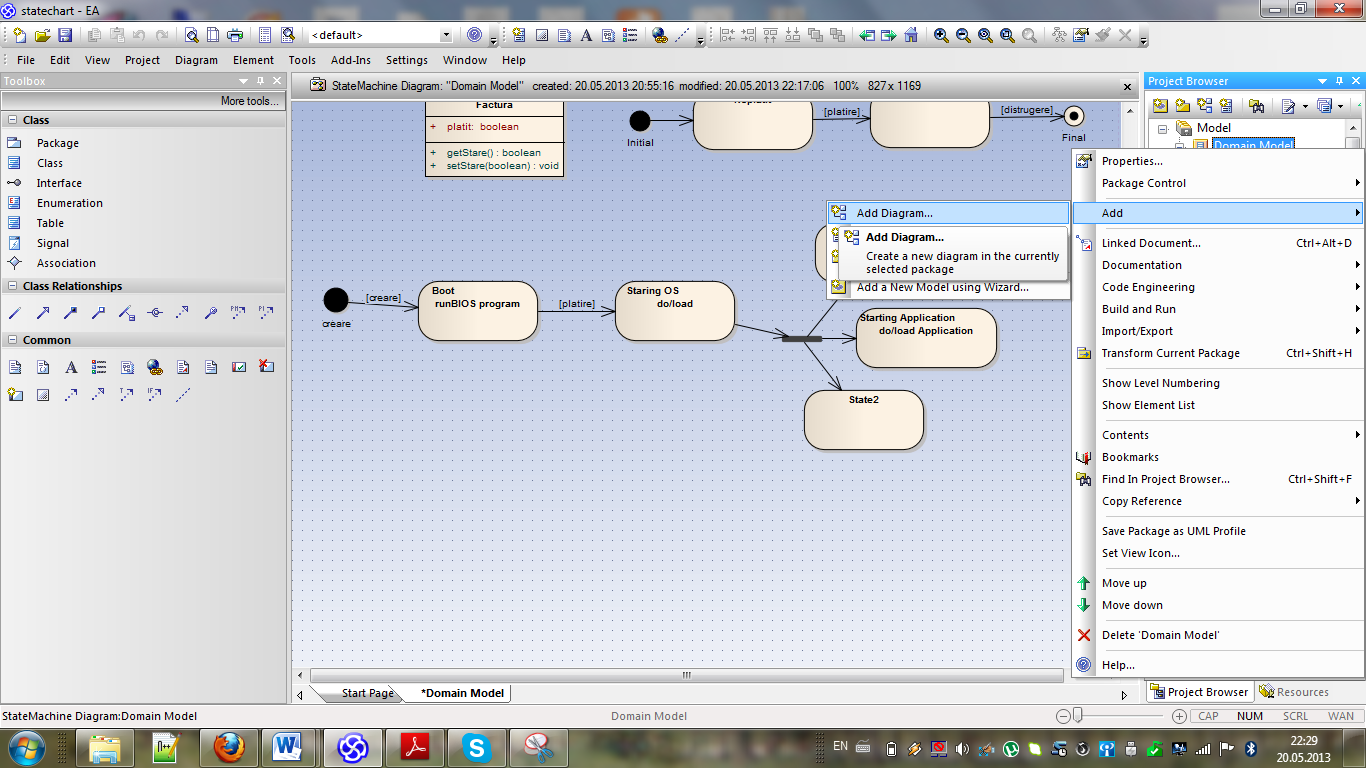
1. Din fereastra *New Project,* selectăm directoriul în care va fi salvat proiectul și edităm denumirea în câmpul *File Name ( Имя файла ),* tipul fișierului, implicit este \*.EAP.



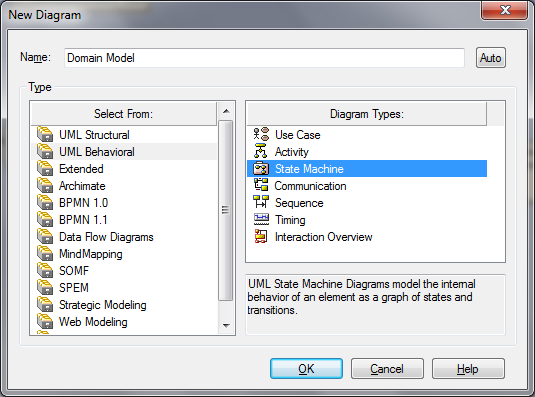
3.Din fereastra *Select model( s )*, alegem valoarea < all > din lista *Select from.* Din lista *Technology* alegem *< default >,* iar din *Name* bifăm *Domain Model.*



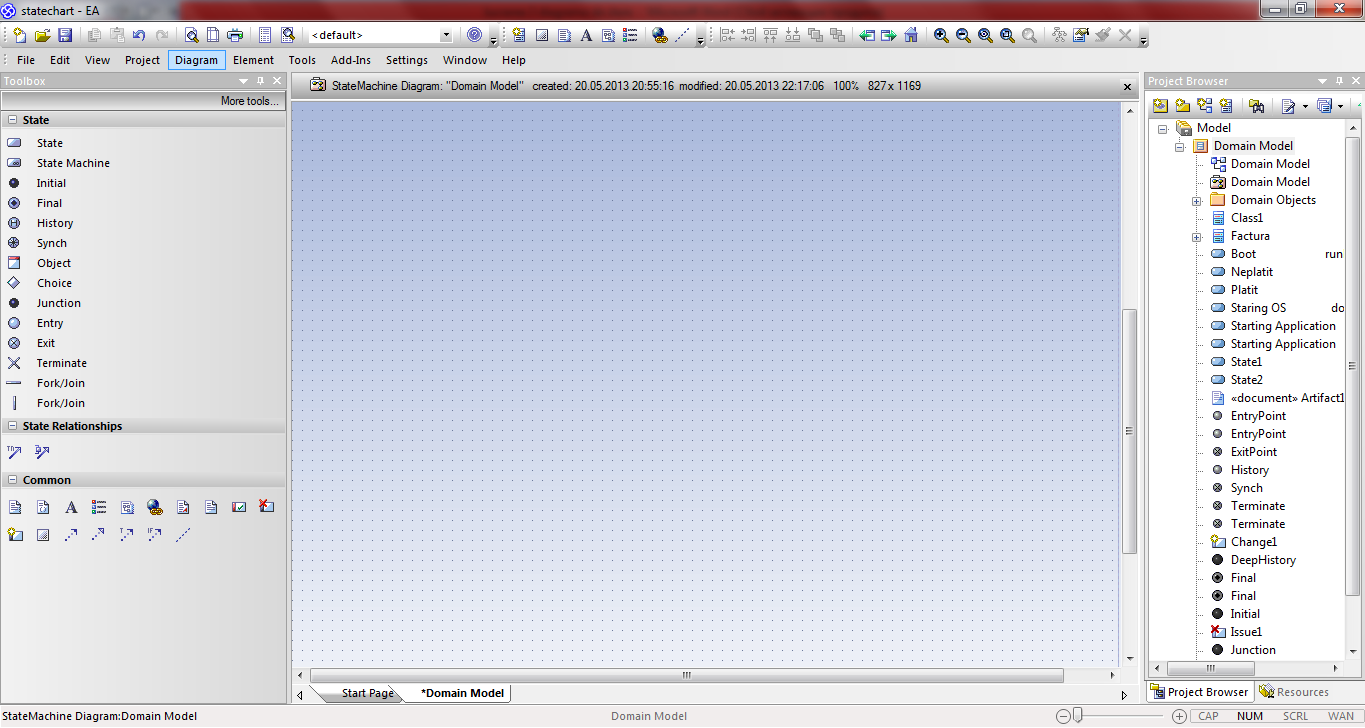
4.Din fereastra *Project Browser* selectăm modelul *Domain Model* și deschidem meniul contextual. Din meniul contextual selectăm comanda *Add -> Add diagram…*



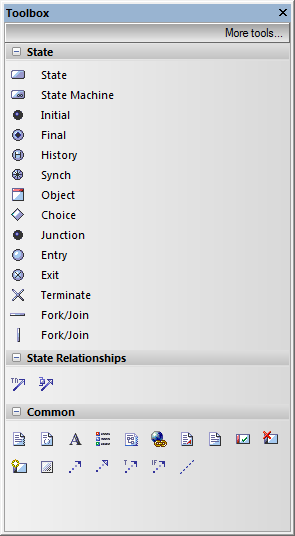
5.Din fereastra *New Diagram* selectăm tipul *UML Behavioral* , iar din *Diagram Types* selectăm tipul diagramei *State Machine*.



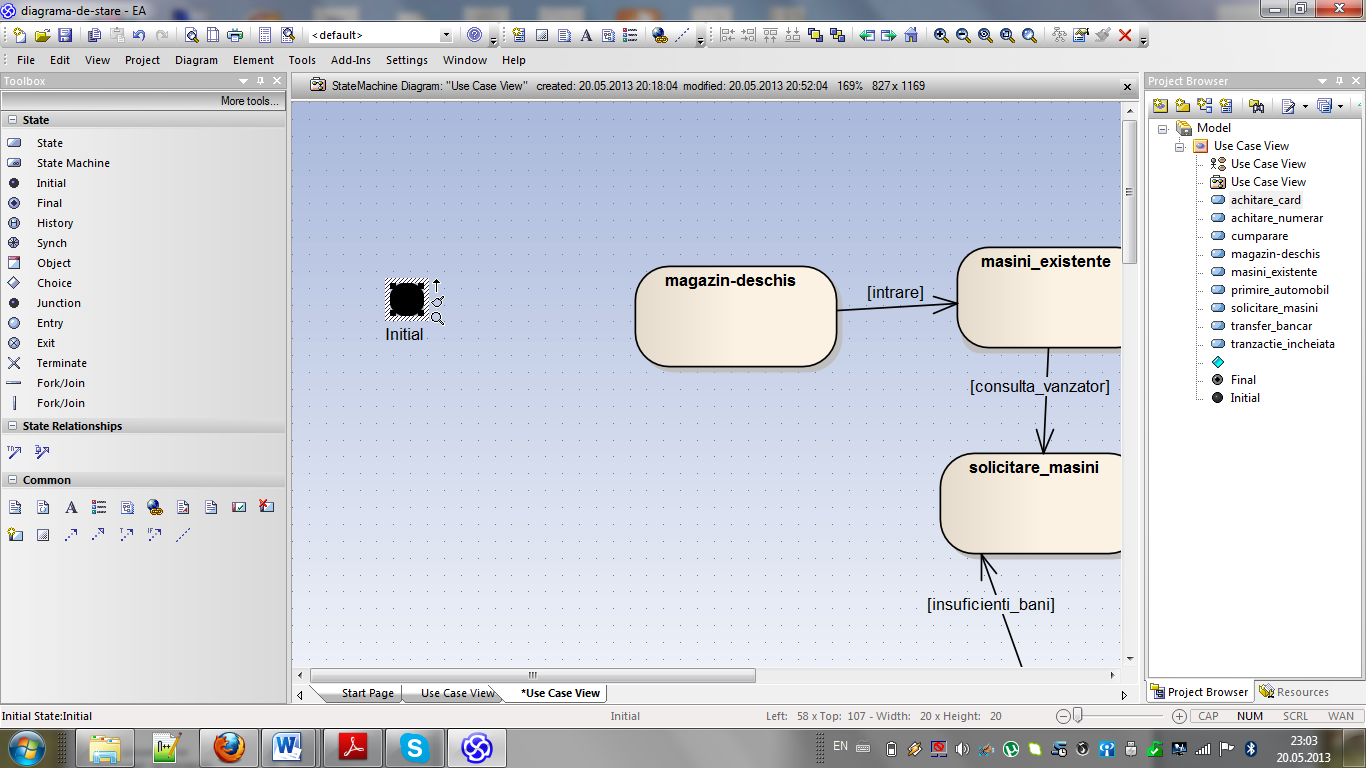
6.Va fi deschisă o fereastră cu suprafață de modelare a diagramei.



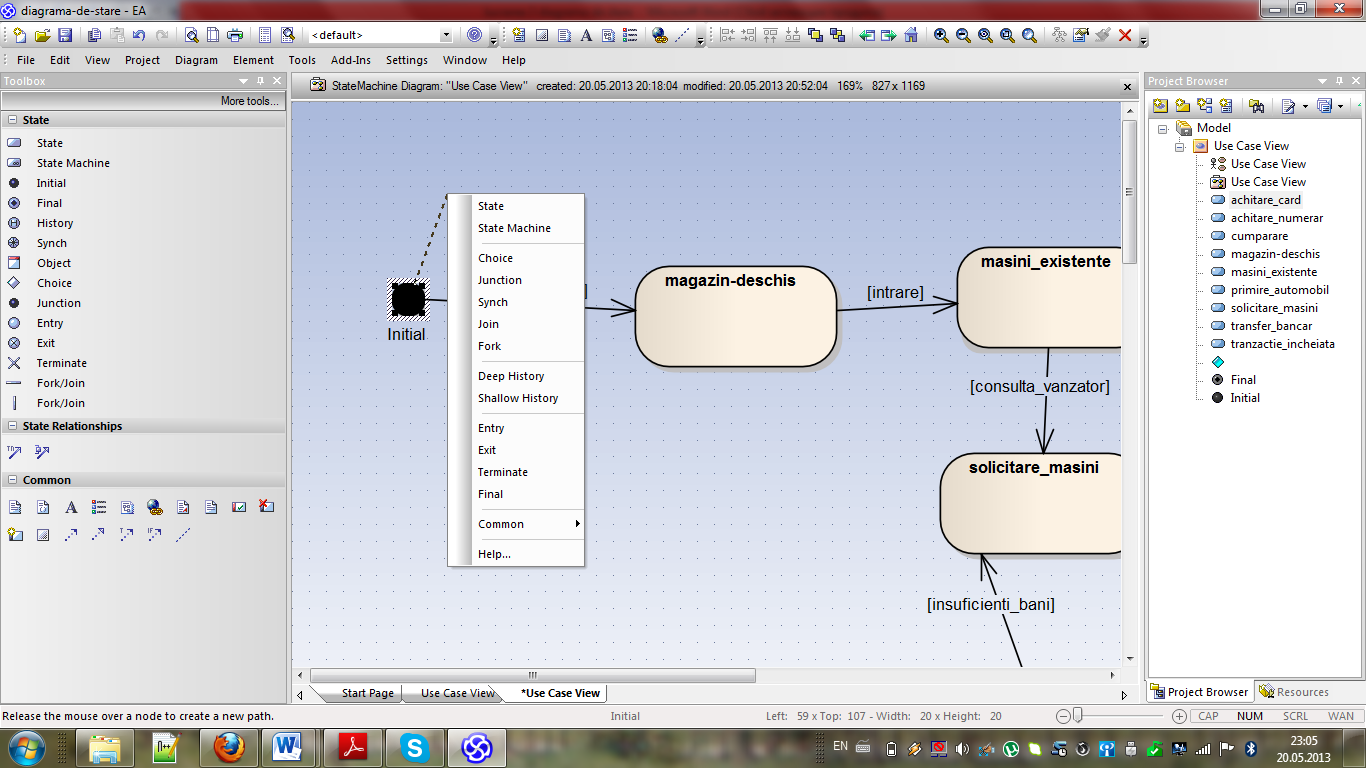
7.Pentru reprezentarea diagramei de stare din bara de unelte ( afișată cu comanda *View - > Toolbox* ) se face clic pe *More tools…* și alegem *State*.

8.Bara de instrumente conține unelte pentru reprezentarea stărilor inițiale ( *Initial* ), finale (*Final* ), fork-urilor, join-urilor, stărilor obișnuite ( *State )* și săgeți pentru reprezentarea tranzițiilor ( în *State Relationships* ).

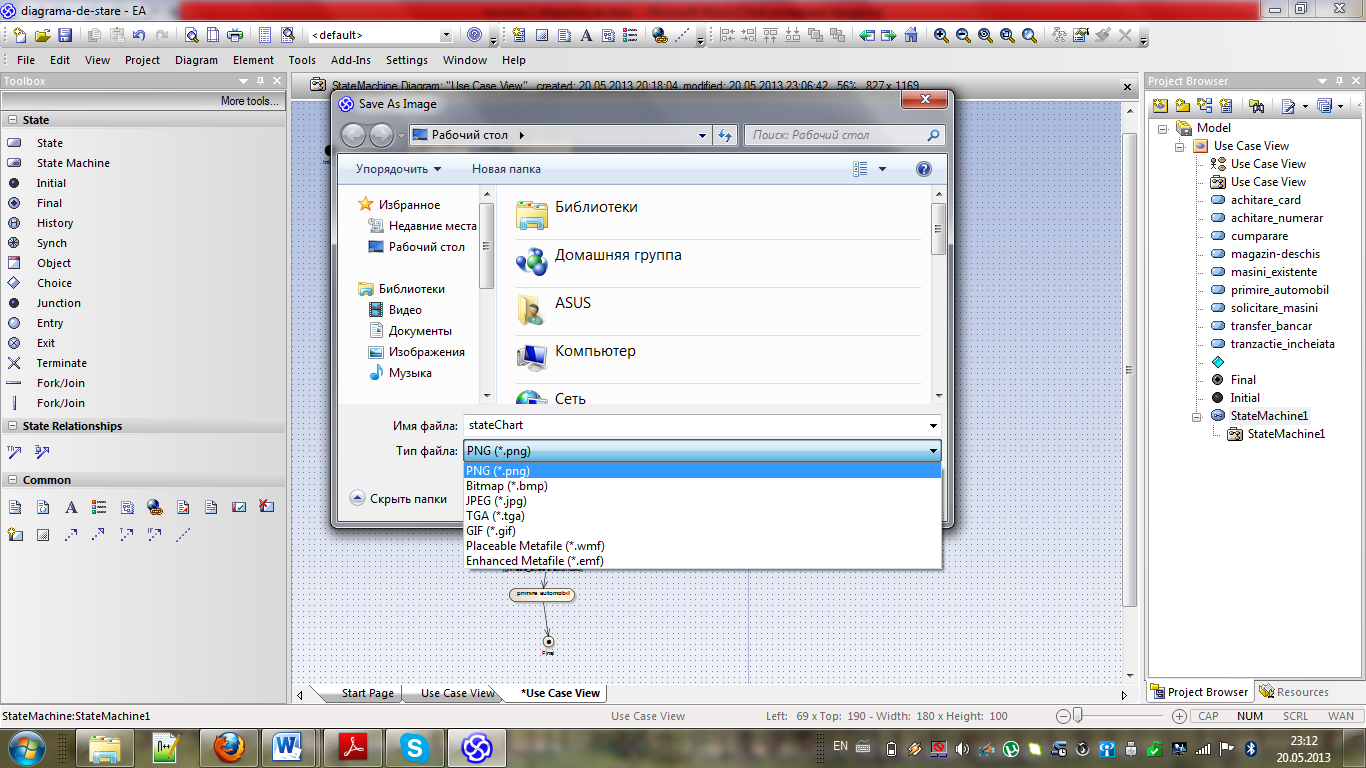
Putem crea stările și apoi să alegem tipurile de relații , adică tranziții pentru a le uni în cadrul diagramei. Altă soluție este selectarea unei stări din diagramă.

Apoi alegem simbolul sub formă de săgeată și îl trasăm pînă va fi deschis un meniu contextual. Din meniu selectăm elementul cu care va fi stabilită legătura sau relația.

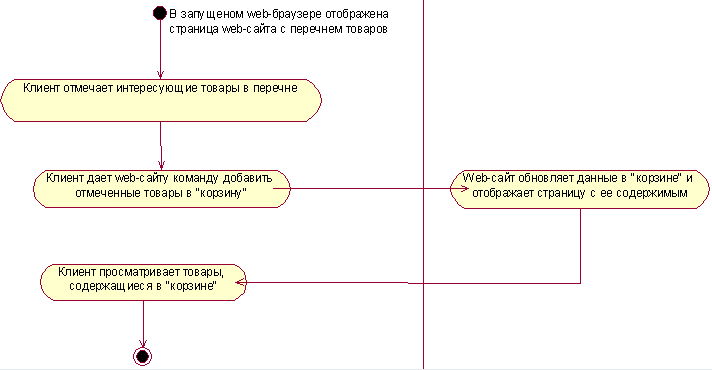
*State –* stare , *State Machine*- mașină de stări, *Choice-* introduce un element de ramificare pentru alegerea unor stări în dependență de unele condiții. *Fork-* element de ramificare, *Join* – element ce unește mai multe tranziții într- una. *Final* – starea finală.



După crearea diagramei o putem salva ca o imagine, selectând comanda *Diagram -- > Save Image…*

Din fereastra deschisă *Save As Image*

selectăm tipul fișierului, denumirea și directoriul unde va fi salvată imaginea.



**Concluzie:**În lucrarea date de laborator am creat diagrama de stare pentru o clasă. Diagrama de clasă poate fi creată și pentru alte clase care prezintă mai multe stări pe parcursul activității sistemului. Acest tip de diagramă este util pentru înțelegerea funcționalității unor părți ale sistemului, arătând stările și tranzițiile de la o stare la alta.

Diagrama de stare arată comportamentul dinamic al claselor ca răspuns la acțiuni din exterior. Din exterior pot fi mai mulți stimuli și stările cauzate de acești stimuli pot fi diferite, dar în diagrama de stare sunt reprezentate doar cele mai importante stări și tranziții în sistemul studiat. Am observat, însă , că pentru a înțelege mai bine această diagramă , trebuie cunoscută și diagrama claselor, pentru a cunoaște atributele, metodele care descriu un obiect al clasei. La fel ar putea fi utilă și diagrama cazurilor de utilizare, pentru a crea stările care să exprime anumite cazuri de utilizare.