**Programmatūras projektējums apraksts (PPA)**

Pēc tam, kad ir izstrādāta un apstiprināta programmatūras prasību specifikācija (PPS), tajā aprakstītās prasības nepieciešams pārnest programmatūras projektējuma aprakstā (PPA), kurā šīs prasības tiek attēlotas atbilstoši programmatūras izstrādes videi.

Projektēšanas laikā tiek pieņemti lēmumi par to, kādā veidā konkrētās prasības tiks realizētas.

Projektējuma apraksta veidošanā var izdalīt vairākus posmus:

* programmatūras arhitektūras projektēšana;
* funkciju projektēšana.

**Programmatūras arhitektūras projektēšana**

Izstrādājot projektējumu, nepieciešams aprakstīt visus veidojamās sistēmas moduļus, definēt, kādas būs moduļu saskarnes un moduļu atkarības.

Par katru vienību ieteicams norādīt tās nosaukumu, nolūku un funkciju.

Projekta programmatūras arhitektūras veidošanas ieteikumi.

Spēles programmatūras projektu varētu sadalīt šādos moduļos:

* tīmekļa lapa un tās elementi,
* kopīgi izmantojamā skripta daļa: mainīgie un funkcijas,
* atsevišķu objektu funkcijas.

Sadalīšana moduļos ļaus arī noteikt tālākos projektēšanas darbus grupās – kurš par kuru moduli atbildēs un ko programmēs.

**Tīmekļa lapas arhitektūra**

Šajā programmatūras izstrādes solī būtu nepieciešams vienoties par lappuses satura galvenajiem elementiem, kas veido saskarni starp spēlētāju un programmu, piemēram:

* lietotājam paredzētie ievades elementi: lodziņi, pogas u.c.
* lietotājam paredzētie izvades elementi: pamatne, lodziņi, teksta rindkopas u.c.

Vajadzētu noteikt lapas elementu un objektu izmērus, ka arī katra ievades un izvades elementa nosaukumu un to, kādu notikumu apstrādes funkcijas šiem elementiem būs nepieciešams veidot un atbilstošo funkcijas nosaukumus. Nosaukumu piešķiršana jau tagad ļaus viennozīmīgi noteikt, par kuru projektējamo vienību tiek runāts.

**Funkciju projektēšana**

Skripta sākumā būs nepieciešams deklarēt mainīgos, kuru datus paredzēts izmantot vairākām funkcijām un koplietojamas funkcijas. Ja arī šajā izstrādes solī konkrētus mainīgos un funkcijas definēt vēl nevar, svarīgi tomēr šādu moduli izdalīt jau tagad.

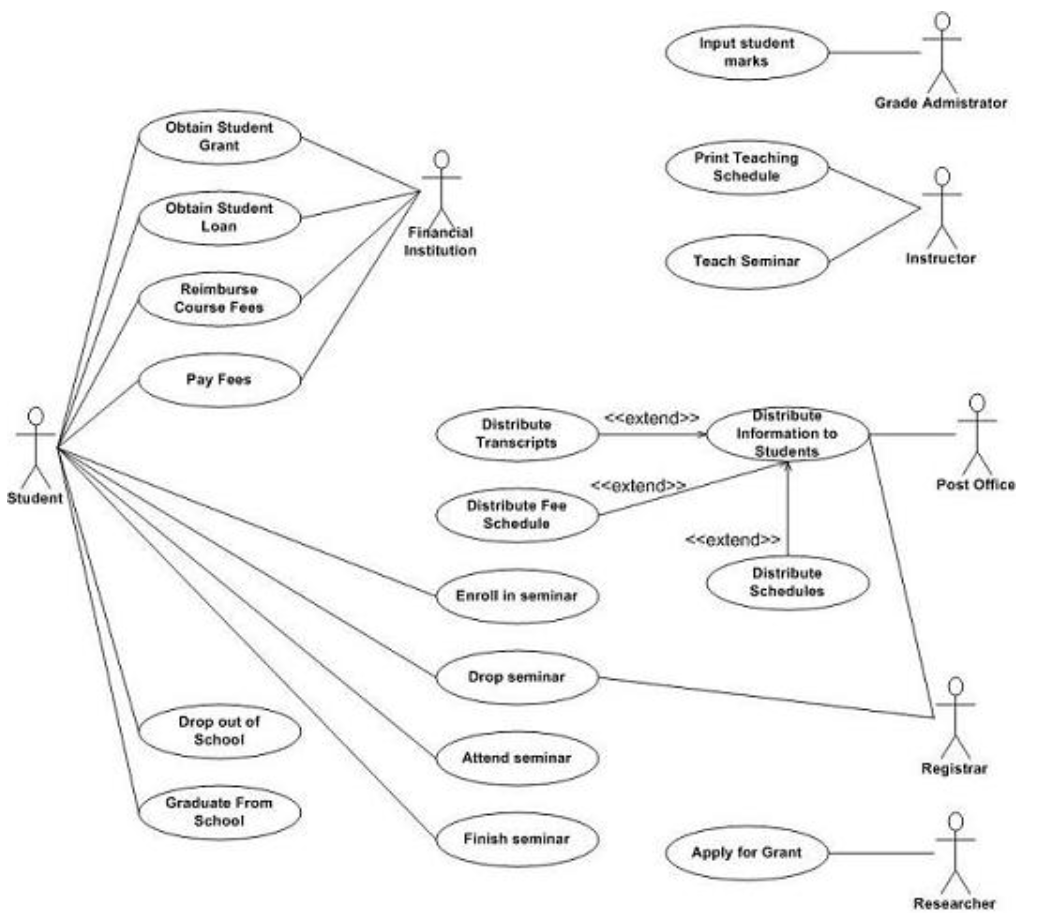
Ka izstrādā spēli, tad spēli parasti veido vairāki objekti, kuri var savstarpēji mijiedarboties. Katram spēles objektam vai objekta veidam (ja ir vairāki identiski) var veidot savu moduli.

**Funkcionālais modelis (Use case model)**

Use case modelis apraksta sistēmas funkcionālo mērķi. Tas ir sistēmas sākotnējais konceptuālais attēlojums. Parasti modelī attēlo visus iespējamos lietotājus un viņas iespējamas darbības sistēmā.

Pakalpojums, kas tiek inicializēts pēc lietotāja pieprasījuma, ir pilnīga darbību secība. Use case īpašības:

* Lietošanas gadījumi apraksta ne tikai mijiedarbību starp lietotājiem un sistēmu, bet arī to, kā sistēma reaģē uz atsevišķu ziņojumu saņemšanu no lietotājiem.
* Lietošanas gadījumi var ietvert pakalpojuma ieviešanas aprakstus un dažādus izņēmumus, piemēram, pareizu sistēmas kļūdu apstrādi.
* Lietošanas gadījuma daudzumam kopumā ir jādefinē visi iespējamie sagaidāmās sistēmas darbības aspekti.

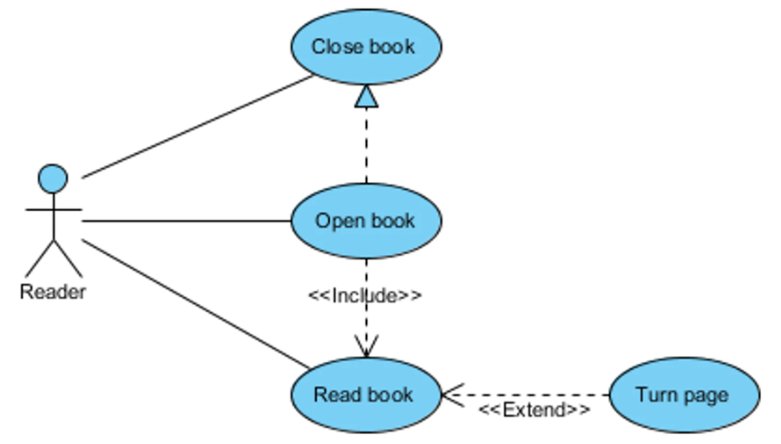
Lietošanas gadījumu piemēri var būt šādas darbības: klienta norēķinu konta stāvokļa pārbaude, preču iegādes pasūtījuma noformēšana, papildu informācijas iegūšana par klienta kredītspēju, grafiskas formas attēlošana ekrānā un citas darbības.

Attēls. Use case piemērs

Mijiedarbības veidi

Starp dalībniekiem un lietošanas gadījumiem var būt dažāda veida mijiedarbība. Galvenie mijiedarbības veidi ir šādi:

* **Vienkārša asociācija** – attēlota ar līniju starp dalībnieku un lietošanas gadījumu (bez bultiņas). Atspoguļo attiecības starp lietotāju un lietošanas gadījumu. Piemēram attēlā starp studenta (Student) un lietošanas gadījuma absolvējis skolu (Graduate From School).
* **Virzīta saistība** – tāda pati ka vienkārša asociācija, taču norāda, ka lietošanas gadījumu inicializē dalībnieks. Apzīmēta ar bultiņu.
* **Mantojums (inheritance)** – norāda, ka pēcnācējs manto savas tiešā priekšteča īpašības un uzvedību. Var attiecināt gan uz lietotājiem, gan lietošanas gadījumiem. Piemēram attēlā starp Post Office lietotāju un Distribute Information to Student. Savukārt, Distribute Information to Student izmanto mantojumu ar 3 lietošanas gadījumiem – Distribute Transcripts, Distribute Fee Schedule, Distribute Schedules.
* **Paplašinājums (extend)** – norāda, ka lietošanas gadījums paplašina darbību pamata secību un ievieto savu secību. Šajā gadījumā, atšķirība no attiecību veida “iekļaušana (include)”, paplašināto secību var veikt atkarībā no noteiktiem nosacījumiem.
* **Iekļaušana (include)** – norāda, ka lietošanas gadījums ir iekļauts bāzes secībā un vienmēr tiek izpildīts.



Attēls. Include un extend lietošanas piemērs