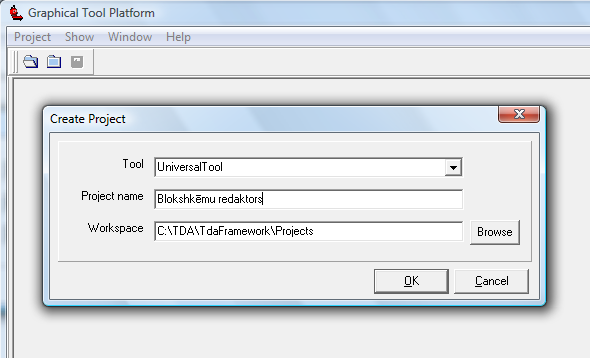
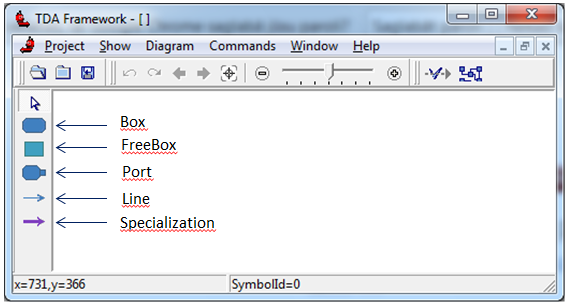
Jauna DSML rīka specificēšana sākas ar jauna platformas projekta izveidi (Project->New project), pēc kuras atveras 4.1. attēlā redzamais dialogu logs.



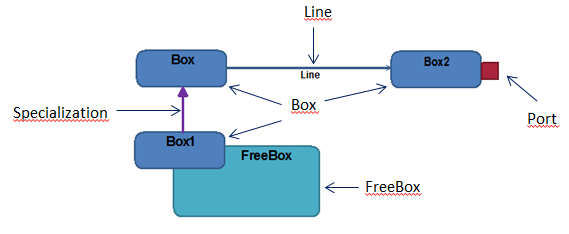
*4.1. att.* Jauna projekta izveidošana

Lai izvēlētos konfiguratoru, laukā *Tool* lietotājam ir jānorāda „UniversalTool”, laukā *Project Name* ir jāievada projekta nosaukums, kas sakritīs ar izstrādājamā rīka nosaukumu, piemēram, „Blokshēmu redaktors”, bet laukā *Workspace* ir jānorāda projekta atrašanās vieta. Pēc jauna projekta izveides atveras projektu diagramma, bet, nospiežot taustiņu „C”, atveras 4.2. attēlā redzamā konfiguratora diagramma.



*4.2. att.* Konfiguratora diagramma

Jaunus elementus konfiguratora diagrammā veido ar paleti, kura sastāv no piecām pogām, un ar katru pogu var izveidot kādu no 4.3. attēlā redzamajiem konfiguratora grafiskajiem elementiem – *Box*, *Line*, *Port*, *FreeBox* un *Specialization*.

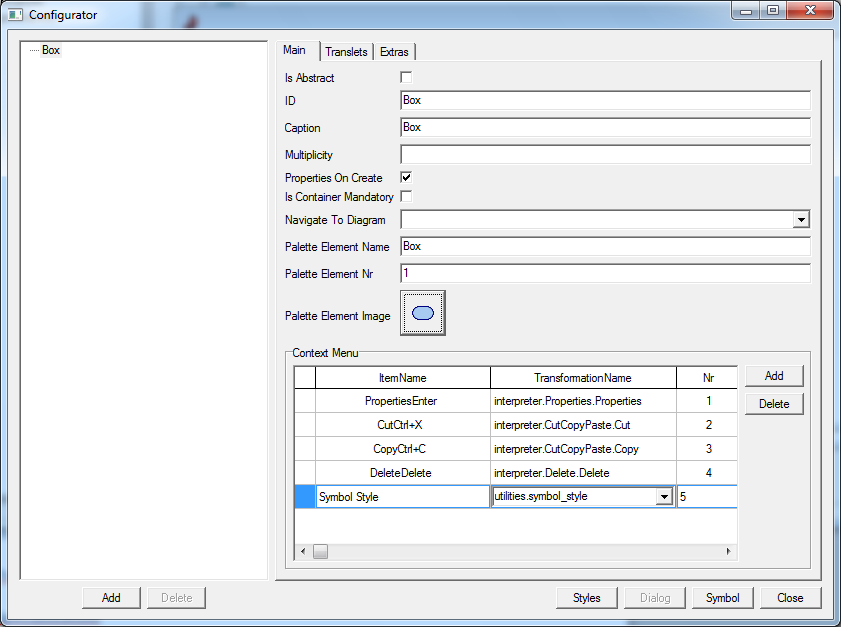


*4.3. att.* Konfiguratora grafiskās valodas elementi

Katrs valodas elements, izņemot *Specialization*, specificē kādu izstrādājamā DSML rīka grafisko elementu, respektīvi, elements *Box* specificē kastes tipa elementus, *Line* specificē līnijas tipa elementus, *FreeBox* specificē brīvās kastes tipa elementus un *Port* specificē porta tipa elementus.

### Box

*Box* elementu veidošana notiek ar paletes elementu *Box* un tā vērtības tiek ievadītas caur dialogu logu, kas ir redzams 4.4. attēlā.



4.4. att. *Box* dialogu loga cilne *Main*

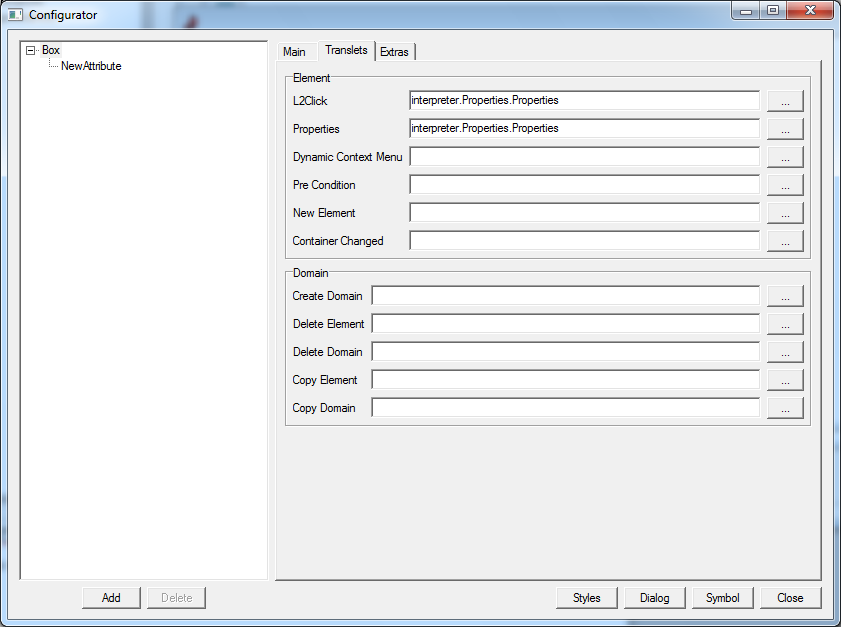
Dialogu loga kreisajā pusē atrodas koks ar elementa un tā atribūtu nosaukumiem (pagaidām neviena atribūta nav), bet labajā pusē atrodas trīs cilnes – *Main*, *Translets* un *Extras*. Cilnes *Main* lauki *ID* un *Caption* ļauj norādīt elementa nosaukumu. Laukā *ID* ir jānorāda elementa unikālais nosaukums un tas drīkst saturēt tikai tādus simbolus, kas ir latīņu alfabēta burti, cipari vai „\_”. Laukā *Caption* ir jānorāda elementa reprezentatīvais nosaukums, un tas drīkst saturēt jebkādus simbolus.

Laukos *Palette Element Name*, *Palette Element Nr* un pogu *Palette Element Image* var ievadīt atbilstošā paletes elementa īpašības – nosaukumu, numuru paletē un paletes ikonas adresi. Ja kādu iemeslu dēļ jaunu elementu nevajag veidot ar paletes elementu, tad lauks *Palette Element Name* ir jāatstāj tukšs un atbilstošais paletes elements netiks izveidots.

Laukā *Open Properties On Element Create* var norādīt, vai pēc jauna elementa izveidošanas nepieciešams atvērt dialogu logu, bet laukā *Is Container Mandatory* var norādīt, ka šī tipa kastēm vienmēr ir jābūt ievietotām kādā kastē. Lai norādītu maksimāli pieļaujamo viena tipa kastu skaitu diagrammā, laukā *Multiplicity* ir jānorāda to ierobežojošais skaits. Ja šis lauks ir atstāts tukšs, tad maksimālais ierobežojums nepastāv. Lauks *Navigate To Diagram* ļauj norādīt, ka, veidojot jaunu kasti, kastei vēl nepieciešams piesaistīt jaunu diagrammu. Piemēram, diagrammas piesaistīšana ir nepieciešama gadījumā, kad elementu detalizē ar diagrammu.

Tabulā *Context Menu* var definēt kastes uznirstošās izvēlnes operācijas. Katra tabulas rindiņa atbilst vienai uznirstošās izvēlnes operācijai, un katra operācija tiek specificēta, ievadot operācijas nosaukumu, izpildāmās transformācijas nosaukumu un tās kārtas numuru uz izvēlnes.

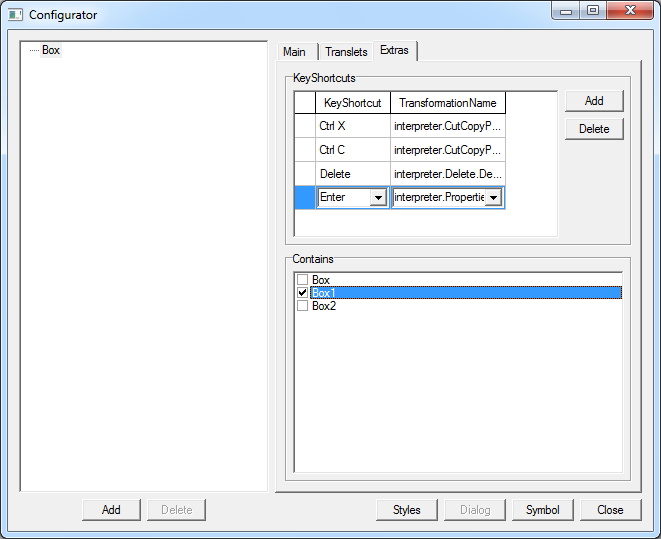
4.5. attēlā ir attēlota *Translets* cilne, kurā ir jānorāda visi tie transleti, kuri kā parametru saņems specificējamo elementu. Šīs cilnes lauku nosaukumi atbilst universālās transformācijas paplašinājuma punktiem (aprakstīti 4. nodaļā), bet to vērtības atbilst transletu nosaukumiem.



4.5. att. *Box* dialogu loga cilne *Translets*

Cilnē *Extras*, kas ir redzama 4.6. attēlā, rīka izstrādātājs var ievadīt īpašības, kuras tiek reti lietotas, vai arī reti mainītas. Cilne sastāv no divām daļām - tabulas *KeyShortcuts* un saraksta *Contains*. Tabulā *KeyShortcuts* var norādīt elementam piekārtotās taustiņu kombinācijas un tām atbilstošās transformācijas, kas tiek izpildītas pēc attiecīgās taustiņu kombinācijas ievades. Katra taustiņu kombinācija atbilst vienai tabulas rindiņai un pēc noklusējuma katrai kastei ir piesaistītas četras taustiņu kombinācijas „Ctrl X”, „Ctrl C”, „Delete” un „Enter” ar tām atbilstošajām transformācijām no transformāciju bibliotēkas, kas atbilst operācijām izgriezt, kopēt, dzēst un atvērt dialogu logu.

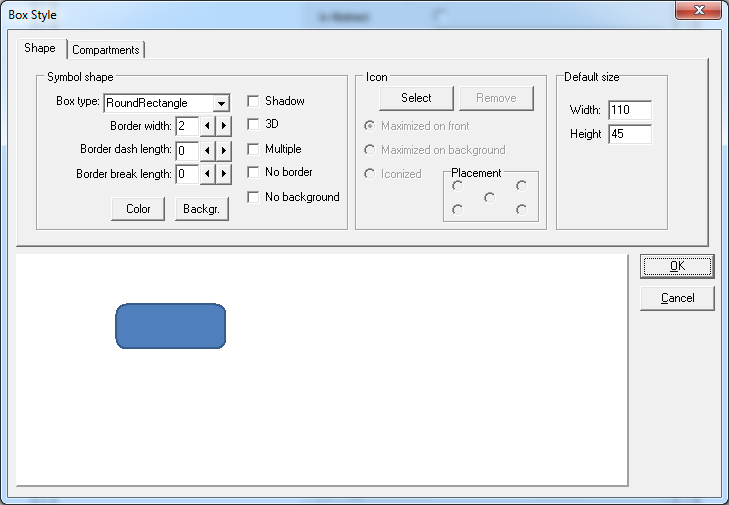
Sarakstā *Contains* ir norādīti visi rīka valodā esošie kastes tipa elementi. Ja kādu saraksta elementu ieķeksē, tad ar to tiek norādīts, ka specificējamā tipa kaste drīkst saturēt ieķeksētā tipa kastes. Piemēram, šajā logā ir norādīts, ka *Box* tipa kastes drīkst saturēt *Box1* tipa kastes.



4.6. att. *Box* dialogu loga cilne *Extras*

Pēc kastes īpašību ievadīšanas ir jāspecificē kastes izskats. To var izdarīt nospiežot pogu *Symbol*, kas atver 4.7. attēlā redzamo dialoga logu. Cilnē *Shape* ir paredzēta kastes stila definēšana un tā ir sadalīta četrās grupās – *Symbol shape*, *Icon*, *Default size* un lauks, kurā var

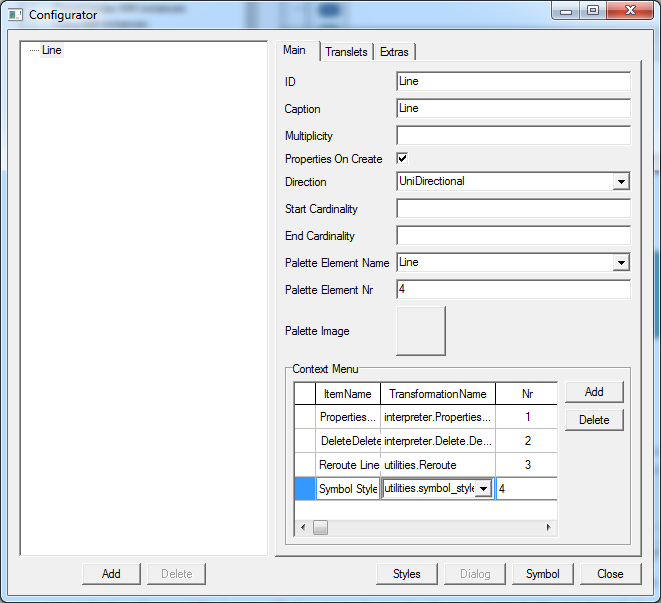
redzēt kastes izskatu. Svarīgākā grupa kastes stila definēšanā ir *Symbol shape*. Šīs grupas lauks *Box type* ļauj norādīt kastes formu, piedāvājot sarakstu ar iepriekš definētām formām, piemēram, taisnstūri, elipsi, trapeci utt. Laukā *Border width* ir jānorāda kasti ietverošā rāmja līnijas biezums, kas pēc noklusējuma ir „1”, bet vispārīgā gadījumā – jo lielākā vērtība, jo platāka līnija. Lauki *Border dash length* un *Border break length* ļauj norādīt, ka ietverošā rāmja līnija ir raustīta. Pēc noklusējuma abu lauku vērtība ir „0”, kas norāda, ka līnija nav raustīta. Ja kādu no šīm vērtībām palielina, tad līnija kļūst raustīta. *Border dash length* ļauj norādīt, cik gara ir raustītā līnijas, bet *Border break length* ļauj norādīt, cik garš ir tukšums uz līnijas. Ar pogu *Color* var uzstādīt ietverošā rāmja līnijas krāsu, bet ar pogu *Backgr*. Var uzstādīt kastes krāsu. Pārējie lauki ļauj uzstādīt dažādus papildus specefektus. Grupa *Icon* ļauj kasei piesaistīt bildi. Ar pogu *Select* var bildi pievienot, bet ar *Remove* noņemt. Piesaistītās bildes pozīciju diagrammā var norādīt *Placement* apakšgrupā kopā ar iezīmētu *Iconized* lauku. Ja ir iezīmēts *Maximized on front*, tad bilde pārklāj visu kasti ar visiem atribūtiem, bet, ja ir iezīmēts *Maximized on background*, tad bilde pārklāj visu kasti, atribūtus atstājot virspusē. Savukārt grupa *Default size* ir paradzēta noklusētā kastes platuma un augstuma uzstādīšanai.



*4.7. att. Box* stila dialogu logs

### Line

Ja izvēlas paletes elementu *Line* un pēc tam savieno divus citus diagrammas elementus, tad ar to tiek pateikts, ka par šī tipa līnijas galapunktiem drīkst būt norādītā tipa elementi, bet elementa īpašības ievada caur 4.8. attēlā redzamo dialogu logu.

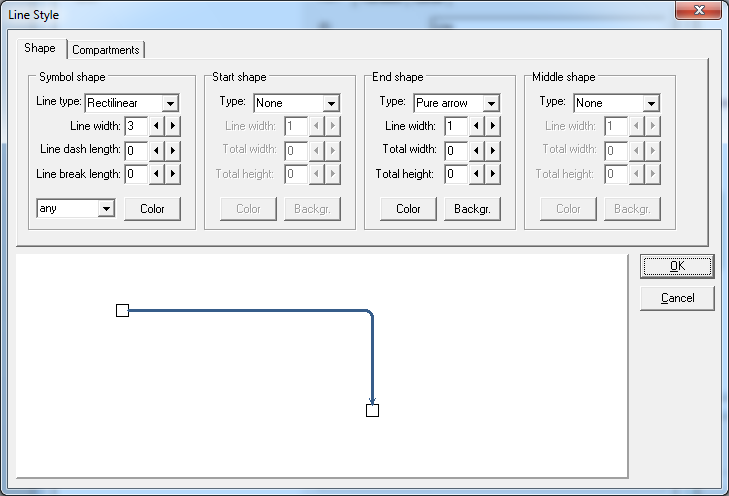


*4.8. att. Line* dialogu loga cilne *Main*

Lai arī *Line* dialogu loga *Main* cilne ir līdzīga ar *Main* cilni *Box* elementam, tām ir vairākas atšķirības. Pirmkārt, nav lauku *Is Container Mandatory* un *Navigate To Diagram*. Otrkārt, ir jauns lauks *Direction*. Šis lauks ļauj norādīt līnijas orientāciju, un pieļaujamās vērtības ir – „UniDirectional”, „BiDirectional” un „ReverseBiDirectional” (tās atbilst 3.1.2. nodaļā skaidrotajām klases *EdgeTpye* atribūta *direction* vērtībām). Treškārt, ir jauni lauki *Start Cardinality* un *End Cardinality*, kas norāda līnijas galu elementiem maksimālo izejošo un ieejošo viena tipa līniju skaitu. Ceturtkārt, *Palette Element Name* lauka tips ir izkrītošā izvēlne, nevis vienkāršs teksta lauks. Izvēlne sastāv no visu valodā esošo līniju paletes elementiem, un, izvēloties kādu no tiem, tiek norādīts, ka ar izvēlēto paletes elementu tiks veidotas arī specificējamā tipa līnijas, respektīvi, ar šo mehānismu ir iespējams norādīt, ka dažādu tipu līnijas tiek veidotas ar vienu paletes elementu. Savukārt, ja laukā ievada izvēlnē neesošu nosaukumu, tad tik izveidots jauns paletes elements.

Cilnēs *Translets* un *Extras* ievadāmā informācija ir līdzīga *Box* dialogu logos ievadāmajai. Cilnē *Translets* var norādīt transletus, kas attiecas uz līnijas paplašinājuma punktiem, bet cilnē *Extras* var norādīt elementam piekārtotās taustiņu kombinācijas.

*Line* stilu definēšanas logs ir redzams 4.9. attēlā. Tā cilne *Shape* sastāv no grupām *Symbol shape*, *Start shape*, *End shape*, *Middle shape* un lauka, kurā var redzēt līnijas izskatu. Grupa *Symbol shape* attiecas tieši uz līnijas stila definēšanu. Laukā *Line type* var norādīt līnijas tipu, laukā *Line width* var norādīt līnijas biezumu, bet laukos *Line dash length* un *Line* *break length* var norādīt, ka līnija ir raustīta, bet ar pogu *Color* var norādīt līnijas krāsu. Grupās *Start shape*, *End shape* un *Middle shape* ir lauks *Type*, kas ļauj norādīt līnijas sākuma, beigu un vidus simbolu, piemēram, bultu, trijstūri, riņķi utt. *Line width* ļauj norādīt simbolu ierobežojošās līnijas biezumu, bet *Total width* un *Total height* norāda simbola platumu un garumu. Poga *Color* ļauj norādīt simbolu ierobežojošās līnijas krāsu, bet *Backgr*. ļauj norādīt simbola krāsu.



*4.9. att. Line* stila dialogu logs

### Port

Ja nospiež paletes pogu *Port* un pēc tam nospiež uz kāda *Box* elementa, tad tiek izveidots jauns *Port* elements, un ar to tiek pateikts, ka jaunajā rīkā tiek izveidots jauns porta tipa elements, kuru drīkst pievienot šī tipa kastei. Tā kā šī elementa īpašībās ievada caur dialogu logu, kura struktūra sakrīt ar citu elementu dialogu logu struktūru, un stilu definēšanas logs idejiski ir vienāds ar *Box* stila logu, tad šos logus detalizētāk neaplūkosim.

### FreeBox

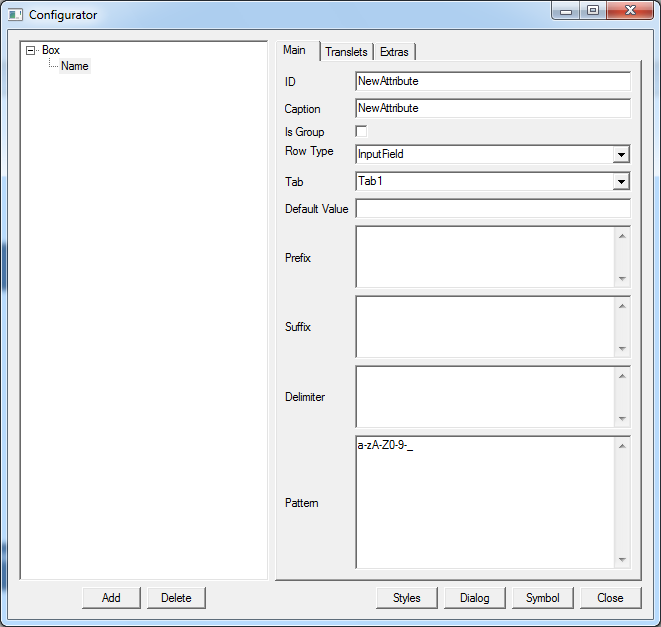
Ja nospiež paletes pogu *FreeBox* un pēc tam izveido jaunu kasti, tad tiek izveidots jauns *FreeBox* elements, un attiecīgi jaunajā rīkā tiek izveidots jauns elements ar brīvās kastes tipu. Arī šī elementa logus neaplūkosim detalizētāk, jo īpašību dialogu loga struktūra un stilu definēšana sakrīt ar *Box* logiem.

### Specialization

Ja nospiež paletes pogu *Specialization* un pēc tam savieno divus diagrammas elementus, tad tiek izveidots jauns *Specialization* elements. Ar šo elementu tiek norādīts, ka viens diagrammas elements ir apakštips otram.

### Atribūtu specificēšana

Atribūtu specificēšana visiem elementiem neatkarīgi no to tipa ir vienāda un tādēļ apskatīsim tikai atribūta specificēšanu kastes gadījumā. Jauni atribūti tiek pievienoti ar pogu *Add*, un tā rezultātā zem iepriekš aktīvās koka virsotnes tiek izveidota jauna koka virsotne, kuru uzreiz arī aktivizē.



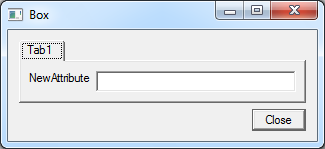
*4.10. att.* Atribūtadialogu loga cilne *Main*

Atribūta vērtības ir jāievada caur 4.10. attēlā redzamo dialogu logu. Līdzīgi kā elementiem, laukā *ID* ir jānorāda elementa unikālais nosaukums un tas drīkst saturēt tikai tādus simbolus, kas ir latīņu alfabēta burti, cipari vai „\_”, bet laukā *Caption* ir jānorāda elementa reprezentatīvais nosaukums, kas drīkst saturēt jebkādus simbolus.

Atribūta dialogu logā tiek norādītas divu veidu īpašības. Vienas specificē atribūta vētību veidošanu, piemēram, noklusēto vērtību vai prefiksu, bet otras specificē informāciju dialogu logu veidošanai. Lauki *Row Type* un *Tab* attiecas uz dialogu logu veidošanu. Laukā *Row Type* ir jānorāda ievadlauka tips atribūta vērtību ievadīšanai. Šim laukam ir izkrītošās izvēlnes tips, kur izvēlne sastāv no 2.1. tabulā uzskaitītajiem ievadlauku nosaukumiem.

Savukārt, ja nepieciešams dialoga loga komponentes grupēt, tad to var izdarīt ar cilnēm. Laukā *Tab* ir jānorāda cilnes nosaukums. Gadījumā, ja norādītais cilnes nosaukums jau eksistē, tad atribūta lauks tiek piesaistīts jau esošajai cilnei. Gadījumā, ja norādītais cilnes nosaukums neeksistē, tad tiek izveidota jauna cilne un atribūta lauks tiek piesaistīts no jauna izveidotajai cilnei.

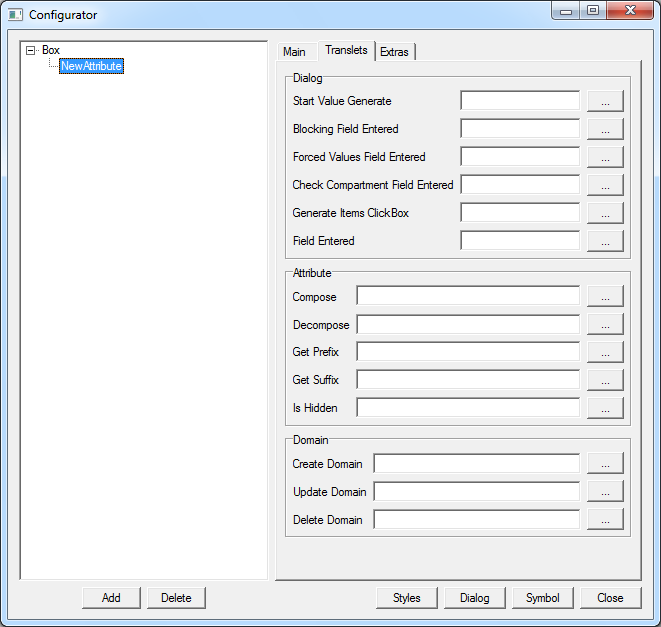
Piemēram, 4.10. attēlā redzamajā dialogu logā ir norādīts, ka atribūta „NewAttribute” ievadlauka tips ir „InputField” un cilnes nosaukums ir „Tab1”. Šai specifikācijai atbilstošais dialogu logs ir redzams 4.11. attēlā.



*4.11. att.* Specificētais dialogu logs

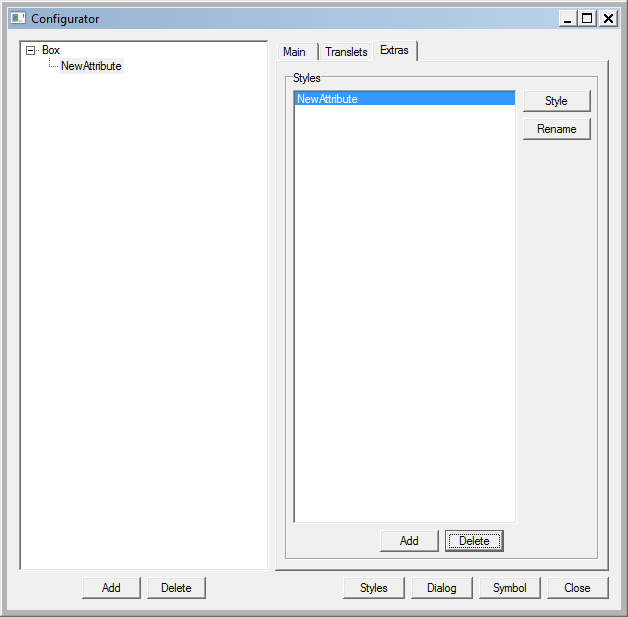
Atlikušo lauku vērtības atbilst klases *CompartType* atribūtu vērtībām. Laukā *Default Value* ir iespējams norādīt atribūta noklusēto vērtību, kas metamodelī atbilst klases *CompartType* atribūta *defaultValue* vērtībai, laukos *Prefix* un *Suffix* var norādīt atribūta prefiksa un sufiksa vērtības un tās atbilst atribūtu *prefix* un *suffix* vērtībām, laukā *Delimiter* var norādīt apakšatribūtu atdalītāja vērtību un tā atbilst atribūta *delimiter* vērtībai, laukā *Pattern* var norādīt atribūta vērtībā pieļaujamos simbolus un tas atbilst atribūta *pattern* vērtībai.

Cilne *Translets* ir redzama 4.12. attēlā. Šīs cilnes laukos var norādīt visus atribūtam piekārtotos transletus, t.i., lauku nosaukumi atbilst atribūtu paplašinājuma punktiem, bet to vērtības ir transletu nosaukumi.



*4.12. att.* Atribūtadialogu loga cilne *Translets*

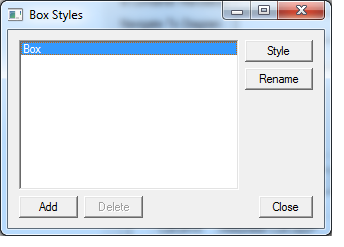
Cilne *Extras* ir redzama 4.13. attēlā. Šajā cilnē atribūtam var piesaistīt jaunus stilus (jaunu stilu pievienošana detalizētāk ir aprakstīta 4.1.7. nodaļā).



*4.13. att.* Atribūtadialogu loga cilne *Extras*

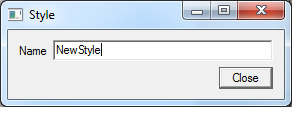
### Stilu pievienošana

Katram elementam pēc noklusējuma ir piesaistīts tieši viens elementa stils. Gadījumā, ja ir vajadzība pēc vairākiem stiliem, tad elementa dialogu logā ir jānospiež *Styles* poga. Tā rezultātā parādīsies 4.14. attēlā redzamais logs.



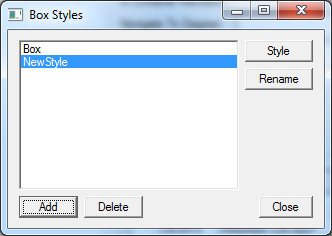
*4.14. att.* Stila specificēšanas logs

Lai pievienotu jaunu stilu, šajā stilā ir jāspiež poga *Add* un tad parādīsies 4.15 attēlā redzamais logs, kurā ir jānorāda stila nosaukums



*4.15. att.* Stila nosaukuma ievade

Stāvoklis pēc stila pievienošanas ir redzams 4.16. attēlā.



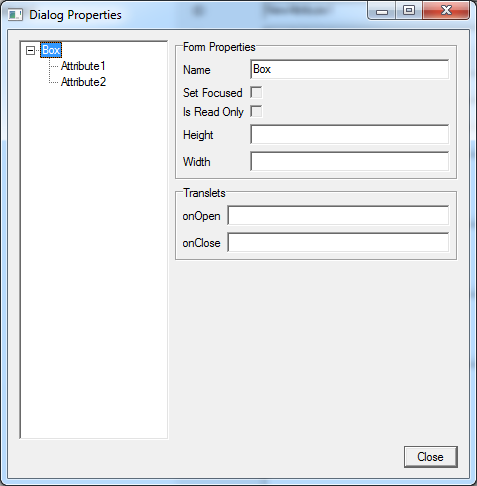
*4.16. att.* Stila specificēšanas logs pēc stila pievienošanas

Lai uzstādītu stila parametrus, ir jānospiež poga *Style* un tad parādīsies elementa tipam atbilstošais dialogu logs, piemēram, ja elements ir *Box*, tad parādīsies 4.7. attēlā redzamais logs, vai, ja elements ir *Line*, tad parādīsies 4.9. attēlā redzamais logs.

### Dialogu logu konfigurēšana

Ja nospiež pogu *Dialog*, parādās 4.17. attēlā redzamais dialogu logs. Dialogu loga kreisajā pusē ir koks, kura saturs atbilst specificētā elementa dialogu loga struktūrai. Piemēram, attēlā redzamā dialogu loga koks reprezentē formu ar diviem laukiem „Attribute1” un „Attribute2”.

Sākotnējā dialogu logu komponenšu secība atbilst to veidošanas secībai, bet, ja nepieciešams secību mainīt, to var izdarīt, ar peli mainot koka virsotnes pozīciju viena vecāka ietvaros.

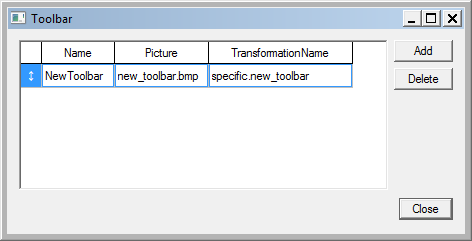


*4.17. att.* Dialogu loga konfigurēšanas logs

Dialogu loga labajā pusē var norādīt aktīvās dialogu loga komponentes īpašības. Piemēram, attēlā redzamajā logā aktīvā komponente ir forma. Grupā *Form Properties* var mainīt dažādus komponentes parametrus, bet grupā *Translets* var norādīt komponentes transletus, kas atbilst 3.2. tabulā aprakstītajam notikumam.

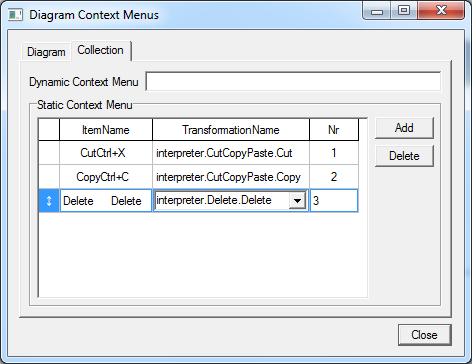
### Diagrammas funkcionalitāte

Nospiežot peles labo pogu uz tukšas vietas diagrammā, parādās uznirstošā izvēlne, kuras operācijas – *Add Toolbar*, *Add Context Menu* un *Add Key Shortcuts* specificē diagrammas funkcionalitāti. Ja no izvēlnes izvēlas operāciju *Add Toolbar*, tad parādās 4.18. attēlā redzamais dialogu logs. Šajā logā katra tabulas rinda specificē vienu rīku joslas pogu. Piemēram, attēlā redzamajā logā ir specificēta rīku josla ar vienu pogu „NewToolbar”.



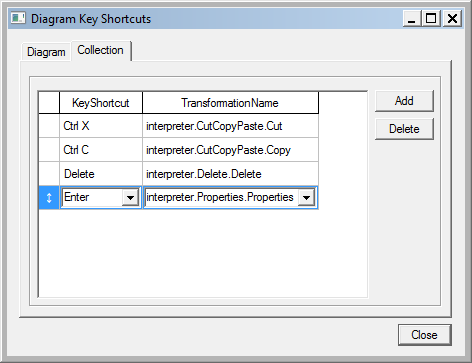
*4.18. att.* Rīku joslas specificēšanas logs

Ja no izvēlnes izvēlas operāciju *Add Context Menu*, tad parādās 4.19. attēlā redzamais logs. Šajā logā var specificēt diagrammas tipa uznirstošās izvēlnes un tiek izšķirtas divas situācijas. Vienā situācijā uznirstošā izvēlne tiek specificēta tukšai kolekcijai un šai situācijai atbilst cilne *Diagram*, otrā situācijā uznirstošā izvēlne tiek specificēt vairāku elementu kolekcijai un šai situācijai atbilst cilne *Collection*. Attēlā redzamajā logā ir specificēta statiskā uznirstošā izvēlne ar trīs operācijām vairāku elementu kolekcijai, bet, lai specificētu dinamisku uznirstošo izvēlni, laukā *Dynamic Context Menu* ir jānorāda atbilstošais translets.



*4.19. att.* Uznirstošās izvēlnes specificēšanas logs

Ja no izvēlnes izvēlas operāciju *Add Key Shortcuts*, tad parādās 4.20. attēlā redzamais dialogu logs. Šajā logā var specificēt diagrammai piesaistītās taustiņu kombinācijas un arī šajā gadījumā tiek izšķirtas divas situācijas. Vienā gadījumā taustiņu kombinācijas tiek specificētas tukšai kolekcijai un šai situācijai atbilst cilne *Diagram*, otrā gadījumā taustiņu kombinācijas tiek specificētas vairāku elementu kolekcijai un šai situācijai atbilst attēlā redzamā cilne *Collection*, kura specificē trīs taustiņu kombināciju apstrādi.



*4.20. att.* Taustiņu kombināciju specificēšanas logs

### Blokshēmu redaktora konfigurēšanas piemērs

Šajā nodaļā aplūkosim, kā ar konfiguratoru var specificēt to pašu blokshēmu redaktoru, kurš tika specificēts ar tipu instancēm 2.3.13. nodaļā. Blokshēmu redaktoru specificēšana sākas ar jauna platformas projekta izveidi, kas tiek izdarīts norādot attiecīgo informāciju 4.1. attēlā redzamajā dialogu logā. Pēc tam, nospiežot taustiņu „C”, atveras konfiguratora diagramma, kurā ir jāspecificē blokshēmu redaktors.

Detalizētāk apskatīsim „Aktivitāte” specificēšanu. Lai specificētu „Aktivitāte”, pirmajā solī ir jāizveido *Box* elements. Tā rezultātā atveras *Box* dialogu logs (4.4. attēls), kurā ir jāievada elementu „Aktivitāte” specificējošās vērtības. Laukā *ID* ir jāievada vērtība bez latviešu burtiem - „Aktivitate”, bet laukos *Caption* un *Palete Element Name* ir jāievada „īstais” nosaukums - „Aktivitāte”. Laukā *Palette Element Nr* ir jānorāda paletes elementa numurs (piemēram, „2”), bet laukā *Palette Element Image* paletes ikonas adrese.

Redaktora specificēšanas otrajā solī ir jāpārslēdzas uz cilni *Translets* (4.5. attēls) un laukā *Dynamic Context Menu* jānorāda transleta nosaukums – „specific.context\_menu\_for\_aktivitate”, ar kuru tiks dinamiski veidotas „Aktivitāte” uznirstošās izvēlnes. Šis translets atrodas failā ar nosaukumu „specific”, bet pats fails atrodas projekta direktorijas apakšdirektorijā „lua”. Transleta kods ir redzams 4.21. attēlā.

|  |
| --- |
| function context\_menu\_for\_aktivitate(elem)  local target\_diagram = elem:find("/target")  local pop\_up\_diagram = create\_pop\_up\_diagram()  if target\_diagram:is\_not\_empty() then  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Dialoga Logs Enter", procedureName = "interpreter.Properties.Properties"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Izgriezt Ctrl+X", procedureName = "interpreter.CutCopyPaste.Cut"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Kopēt Ctrl+C",procedureName = "interpreter.CutCopyPaste.Copy"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Dzēst Delete", procedureName = "interpreter.Delete.Delete"}  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Navigēt", procedureName = "utilities.navigate"})  add\_pop\_up\_element(pop\_up\_diagram,{caption = "Likvidēt Detalizāciju", procedureName = "utilities.remove\_target\_diagram"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Simbola Stils", procedureName = "utilities.symbol\_style"})  else  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Dialoga Logs Enter", procedureName = "interpreter.Properties.Properties"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Izgriezt Ctrl+X", procedureName = "interpreter.CutCopyPaste.Cut"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Kopēt Ctrl+C",procedureName = "interpreter.CutCopyPaste.Copy"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Dzēst Delete", procedureName = "interpreter.Delete.Delete"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Detalizēt", procedureName = "specific.add\_navigation"})  add\_pop\_up\_element(pop\_up\_diagram, {caption = "Simbola Stils", procedureName = "utilities.symbol\_style"})  end  show\_pop\_up\_diagram(pop\_up\_diagram, elem)  end |

*4.21. att.* Dinamiskās uznirstošās izvēlnes kods

Translets, kā parametru saņem aktivitātei atbilstošo objektu, uz kura tika izsaukta uznirstošā izvēlne. Pēc tam translets noskaidro, vai aktivitāte ir detalizēta. Ja aktivitāte ir detalizēta, tad izveido uznirstošo izvēlni ar operācijām – atvērt dialogu logu, izgriezt, kopēt, dzēst, navigēt uz detalizēto diagrammu, likvidēt detalizēto diagrammu un mainīt stilu. Visas šīs operācijas ir realizētas ar transformācijām no transformāciju bibliotēkas (skat. 1. pielikumu), un šajā gadījumā neko papildus nevajag programmēt.

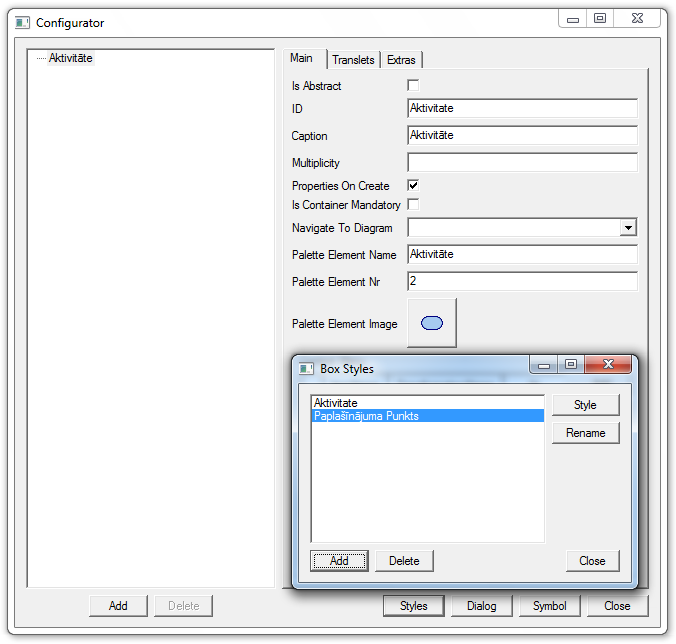
Gadījumā, ja aktivitāte nav detalizēta, tad papildus esošajām operācija tiek pievienota operācija detalizēt aktivitāti, bet noņemtas operācijas navigēt uz detalizēto diagrammu un likvidēt detalizēto diagrammu. Tātad, ja aktivitāte nav detalizēta, uznirstošā izvēlne sastāv no šādām operācijām - atvērt dialogu logu, izgriezt, kopēt, dzēst, detalizēt aktivitāti un mainīt stilu. Aktivitātes detalizēšanas operācija ir realizēta ar šim rīkam specifiski izstrādātu transformāciju, kuras kods ir redzams 4.22. attēlā.

|  |
| --- |
| function add\_navigation()  local elem = utilities.active\_elements()  utilities.add\_navigation\_diagram(elem)  utilities.navigate(elem)  end |

*4.22. att.* Aktivitātes detalizēšanas transformācijas kods

Aktivitātes detalizēšanas transformācija sastāv no trīs soļiem. Pirmais solis ir sameklēt aktivitāti, uz kuras uznirstošā izvēlne tiek izsaukta, otrais solis ir aktivitāti detalizēt jeb pievienot detalizēto diagrammu un trešais solis ir atvērt detalizēto diagrammu. Jāpiezīmē, ka katru soli realizē kāda transformāciju bibliotēkas transformācija, kas vienkāršo specifiskās transformācijas izstrādi.

Lai specificētu „Aktivitāte” noklusēto stilu, jānospiež poga *Symbol*. Tā rezultātā parādīsies 4.7. attēlā redzamais dialogu logs, un tajā jānorāda attiecīgie stila parametri. Blokshēmu redaktorā, kas ir izmantots 4. nodaļā, elementam „Aktivitāte” ir divi stili. Viens ir elementa noklusētais stils (specificēts nospiežot pogu *Symbol*), un to uzstāda jauna elementa radīšanas laikā, bet otrs ir aktivitātes stils, kad tā atbilst paplašinājuma punktam. Lai šo otro stilu specificētu, elementa dialogu logā ir jānospiež poga *Styles*, un tad atvērsies elementa stilu logs (4.14. attēls), kurā jāpievieno jauns stils ar nosaukumu „Paplašinājuma Punkts”. „Aktivitātes” logs pēc stila specificēšanas ir redzams 4.23.attēlā.

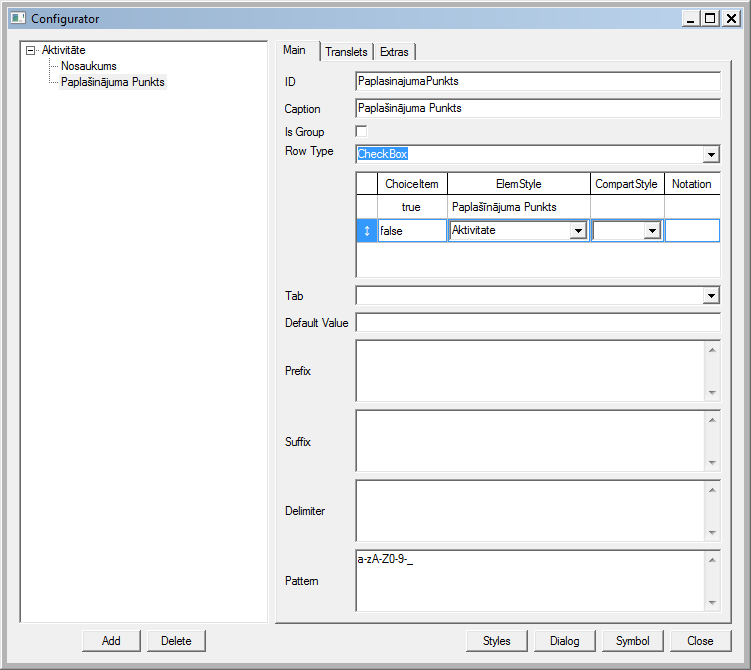


*4.23. att.* „Aktivitātes” logs pēc stila specificēšanas

Pēc „Aktivitāte” funkcionalitātes un stilu uzstādīšanas, ir jāspecificē elementa „Aktivitāte” atribūti „Nosaukums” un „Paplašinājuma Punkts”. Lai specificētu „Nosaukums”, ir jānospiež poga *Add*, un tā rezultātā parādīsies atribūta dialogu logs (4.10. attēls), kura laukos *ID* un *Caption* jāievada vērtība „Nosaukums”, bet laukā *Row Type* jāizvēlas vērtība „InputField”.

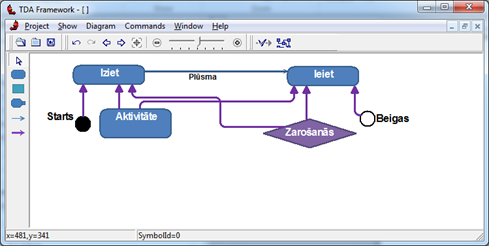
Atribūta „Paplašinājuma punkts” dialogu logs ir redzams 4.24. attēlā. Lai pievienotu atribūtu „Paplašinājuma Punkts”, laukā *ID* ir jāievada vērtība „PaplasinajumaPunkts”, bet laukā *Caption* „Paplašinājuma Punkts”. Savukārt laukā *Row Type* jāizvēlas vērtība „CheckBox”. Pēc tam parādīsies tabula ar divām rindām. Pirmā rinda specificē atribūta vērtības, kad ķekša kaste ir ieķeksēta, otra rinda specificē vērtības, kad ķekša kaste nav ieķeksēta. Viena no šīs tabulas kolonnām ir *ElemStyle*, un šīs kolonnas šūnās ir jānorāda, kāds ir „Aktivitāte” stils, kad ķekša kaste ir ieķeksēta, un kāds ir stils, kad ķekša kaste nav ieķeksēta. Tātad, kad ķekša kaste nav ieķeksēta, ir jānorāda elementa noklusētais stils jeb stils ar nosaukumu „Aktivitāte”, bet, kad ir ieķeksēta, jānorāda stils ar nosaukumu „Paplašinājuma Punkts”.

Savukārt kolonnas *Notation* šūnās jānorāda, kāda ir atribūta reprezentatīvā vērtība gadījumā, kad ķekša kaste ir ieķeksēta, un gadījumā, kad tā nav ieķeksēta. Šajā gadījumā abas vērtības ir „”, un tas nozīmē, kad atribūta „Paplašinājuma Punkts” vērtības diagrammā netiks rādītas, bet, ieķeksējot ķekša kastē, mainīsies aktivitātes stils.



*4.24. att.* „Paplašinājuma punkts” specificēšanas logs

Pārējo blokshēmas redaktora elementu specificēšana ir vienkāršāka, jo priekš tiem nav jāizstrādā papildus transleti un jārada stili, kuri tiek mainīti atkarībā no ķekša kastes vērtībām. Nedaudz atšķirīgāka ir abstrakto elementu „Iziet” un „Ieiet” veidošana. Šiem elementiem ir jāieķeksē laukā *Is Abstract*, lai izdzēstu to paletes elementus un tādējādi padarītu tos abstraktus. Savukārt, lai norādītu, ka abstraktie elementi ir blokshēmas „īsto” elementu virstipi vēl papildus, ir jānovelk *Specialization* līnijas starp abstraktajiem un blokshēmas „īstajiem” elementiem. Rezultātā iegūtā blokshēmu redaktora konfigurācijas grafiskais modelis ir redzams 4.25. attēlā.



*4.25. att.* Blokshēmu redaktora konfigurācija