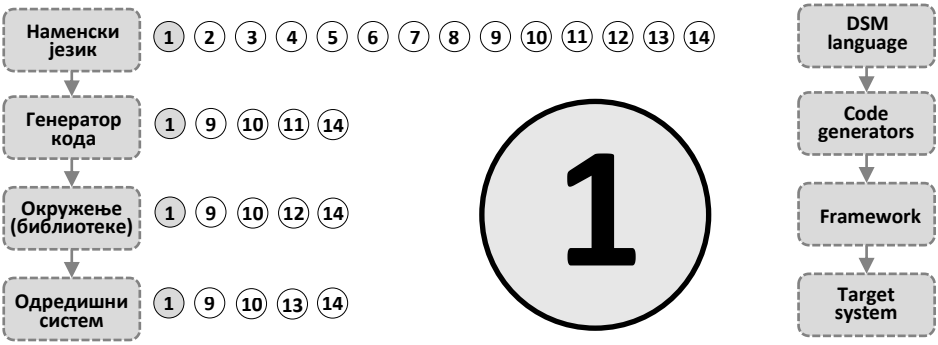
Увод у моделовање помоћу наменских графичких језика



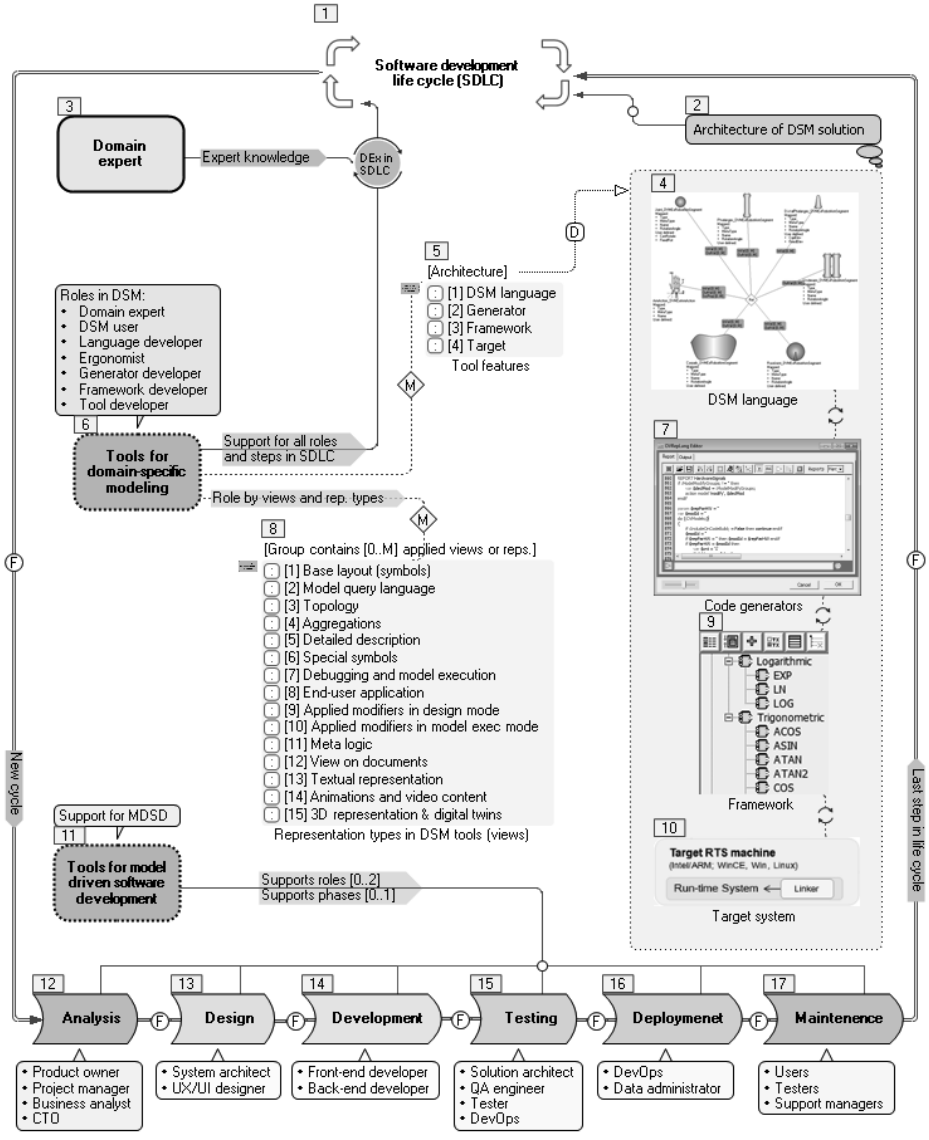
У овом поглављу научићемо шта су то наменски графички језици и које је њихово место у архитектури софтверских решења за моделовање у роботици. Приказано је како се они праве и користе, које проблеме решавају, каква је практична корист од њихове употребе и које предности имају у односу на графичке и програмске језике опште намене. Наменски језици се праве за потребе решавања специфичних, уже стручних проблема и нису исто што и програмски језици опште намене. Употребљивост наменских језика највише зависи од креативности и искуства људи који их конструишу, као и од софтверских алата помоћу којих се тај посао ради. Код наменских графичких језика није битно да паметите њихову синтаксу и семантику, то ефикасно раде анализатори модела, на основу правила која су описана у мета–моделима. Графички језици могу да имају бројне варијације, посебно синтаксе, али то не компликује производњу софтвера. Напротив, сваку варијацију софтвера имамо формално описану у моделима, што нас растерећује читања изворног кода и читања неке друге, мање илустративне и мање актуелне документације. Пажњу треба усмерите на методологију конструкције језика и алата, при чему се на посебан начин узима у обзир област примене и променљивост, као важно својство језика. Овде нема унификације ни унификованих језика. Специфичност језика, коју диктира експертско знање, основно је својство које нам помаже да решимо сложене проблеме из неке струке. Успешност сваког посла, па и конструкције и примене језика и алата, зависи од учесника у послу и правилно распоређених улога. Ваш је задатак, као читаоца, да читајући ову књигу, себе истовремено видите као корисника и креатора језика, али и као корисника и креатора алата за моделовање у роботици.

Наменски језици у роботици (енг. Domain-specific languages) претежно су графички језици који се конструишу и користе за потребе једног специфичног робота, типа робота који производи једна фирма, или за потребе неког конкретног посла који робот треба да обави. Ова дефиниција не искључује примену наменских језика на опште, ни општих језика на специфичне проблеме. Специфично знање и технологија не негира опште знање и технологију, већ је унапређује. У неком тренутку начин употребе и представљања експертског знања постају неефикасани, па доменским експертима треба да се омогући једноставнија употреба и представљање знања помоћу језика њихове струке. За нас, инжењере, то нису језици који већ постоје као део неког стандарда, него су то они језици које треба да развијемо у тесној сарадњи са доменским експертима, односно стручњацима из области примене робота. У архитектури решења за доменско моделовање језици се налазе на највишем нивоу у хијерархији (слика 1, ■4). Они су средство за разумевање свих који учествују у конструкцији и примени робота за одређене потребе.

Наменски графички језик за роботику састоји се од тзв. језичких концепата, односно елемената који представљају објекте и својства реалног света робота, од њихових релација и улога у релацијама, као и од бар једне графичке представе за сваки језички концепт (зглоб, рука, сегмент, стање, акција, операција, кретање, правило, прекидач, мотор итд.). Сви елементи језика се сврставају у мета-типове, тако да сваки елемент језика припада неком од следећих мета-типова: објекат, релација, улога, својство (енг. property), порт или модел. Не мора сваки језик за моделовање да има само ове набројане мета-типове. Но, у пракси се показује да се помоћу ових мета-типова могу изразити сва својства конструкције, кретања, операције робота, као и окружења у ком робот обавља задати посао.

Док се у машинској и електронској индустрији прави јасна разлика између конструкције и примене алата, где конструкција значи развој, а примена значи производњу, у софтверском инжењерству се они најчешће поистовећују. Због тога је теже да се разуме сврха конструкције софтверских алата за производњу софтвера, за било које потребе, па и за роботику. Корисно је да се ова аналогија примени и у софтверском инжењерству. На слици 1 приказана су два модела производње софтвера. У првом моделу (■1) софтвер се производи без обавезне употребе наменских алата (■11), иако се они понекад и у неким фазама користе. Интеграција резултата рада са тако одвојеним алатима представља озбиљан проблем. Први модел производње разликује следеће фазе: анализу проблема и корисничког захтева (■12), дизајнирање архитектуре софтвера (■13), развој (■14), тестирање (■15), дистрибуцију и увођење у употребу (■16) и одржавање (■17). Елементи који шематски описују овај модел производње налазе са на контурама слике 1. Посао се одвија по циклусима, чиме се постиже паралелизам њему, али најчешће паралелизам у производњи различитих софтвера. У производњи се ангажују тимови, софтвераши различитог профила. За сваку од фаза везано је по неколико доминантних инжењерских или консултантских улога, које су гаранција да се посао обавља систематично и на највишем могућем нивоу које фирма за производњу софтвера може да обезбеди. Иако овај приступ не искључује примену разних алата (■11), и у посао укључује будуће власнике или наручиоце софтвера (енг. product owner, ■12), не постоји никаква гаранција да ће

се експертско знање из области примене софтвера преточити и у софтверски производ и у алат. То знање ће се у највећем броју случајева преточити у кодирани текст по синтакси неког програмског језика и у документацију, чија ажурност према кодираном тексту и корисничким захтевима постаје све мања како пролази време.



Слика 1: Различити модели производње и развоја софтвера

Побољшање првог модела производње софтвера, који чине елементи по контурама модела (■1, 11, 12-17), састоји се од метода, алата, улога и послова који су смештени у унутрашњем делу слике. Циљ овог другог начина није само производња софтвера, већ и прављење алата за његову производњу (■6). Зато је

архитектура очекиваног решења другачија и заснива се на архитектури тзв. доменски специфичног моделовања - ДСМ (■2). То решење подразумева конструкцију наменских графичких језика (■4), њихову примену и интерпретацију помоћу генератора кода (■7), при чему се користе постојеће или развијају нове библиотеке или окружења (енг. framework) (■9), а са циљем да се добију програми и друге творевине које могу одмах да се извршавају на неком одредишном систему (■10). Све фазе производње софтвера се одвијају применом алата (■6) који подржавају овакав модел развоја и производње. Експерти из области за коју се развија софтвер (■3) постају важан део процеса развоја. Улоге учесника се знатно мењају у односу на први модел. Оне се остварују такође паралелно, али и у оквиру производње једне верзије софтвера за истог наручиоца. Препознатљиве су улоге доменског експерта, корисника алата за моделовање, креатора језика, ергономисте, креатора генератора, библиотека или окружења и алата за моделовање (■6).

Да би се на једном месту и помоћу једног алата подржале све улоге у производњи софтвера за роботику, и према првом и према другом моделу производње, ти алати мора да се унапреде да подрже брзу и једноставну конструкцију различитих погледа, подмодела, варијација семантике и синтаксе, да поједноставе претраживање, генерисање документације, 3Д приказ итд. (■8). Пошто је тешко наћи алат који има све, чак и за веома уску област примене, у књизи је објашњено како треба правити такве алате и каква је практична корист њихове примене са становишта различитих улога у фазама животног циклуса развоја и производње софтвера. Приказани су примери графичких језика за моделовање, методологија развоја таквих језика и методологија развоја алата. Циљ је да читалац стекне јасну представу како треба да изгледа неко развојно окружење за цртање и извршавање модела и свих његових варијација семантике и синтаксе. Осим тога, знатна пажња је посвећена начину превођења модела и језичких елемената са вишег нивоа апстракције у асемблерски код, протоколе, приказе, функције за дијагностику, документацију, паралелне послове, сигнале и разне друге творевине које експертско знање чине формализованим, употребљивим и извршивим.

Опис елемената модела на слици 1 послужиће као увод у методологију конструкције и примене наменских језика независно од роботике. Приликом прављења шеме два модела производње и развоја уведени су неки типови језичких елемената, правила њиховог повезивања, као и различите графичке представе за различите типове елемената. Елементи се разврставају у типове зато што имају нека заједничка својства. Управо је то разврставање елемената језика по типовима важан део конструкције наменских језика за моделовање, који су применљиви на разне варијанте производње и развоја софтвера. У поступку типизације ради се и разврставање елемената на мета-типове: објекте, улоге, релације, портове, својства (пропертије) и графове. Уочавају се и формално описују правила међусобног повезивања и ограничења шта на моделу није дозвољено и шта мора бити задовољено.

Конструкција наменских графичких језика је сложен интелектуални процес који може да се упореди са решавањем најсложенијих лингвистичких проблема. Но, тај проблем постаје знатно једноставнији ако крај себе имате стручњаке који

добро познају свој интерни језик струке. Њихов језик не мора да буде формално описан, али је важно да је довољно богат и прецизан и да се његови корисници добро разуму. У овом случају најближи сарадници су нам експерти за различите моделе производње софтвера, чији језик је описан у наставку. Када будемо прешли на област роботике, ти најближи сарадници ће нам бити експерти за конкретну област примене робота, али и роботичари, аутоматичари, машинци, електроничари и софтвераши.

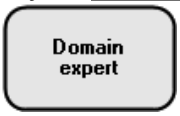
[1] Објекат Review:SoftProdDevModel

Представља један циклус производње софтвера, који почиње јасним, препознатљивим кораком (■12) и завршава се на исти начин (■17). Својства једног циклуса су подаци, као што је нпр. идентификатор верзија кода, или модела.

[2] Објекат noteArch:Annotation

Напомена везана за специфичност архитектуре појединачних решења у развоју и производњи. У овом случају то је напомена о архитектури доменског моделовања, односно решења која се заснивају на конструкцији и примени наменских језика.

[3] Објекат DomainExpert:RoleInDSM



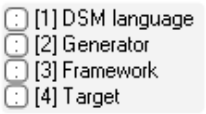
Доменски експерт има различите улоге у зависности од модела производње. Овај симбол означава његову улогу у развоју који се заснива на примени наменских графичких језика. У другом случају он се јавља као власник производа (■12) и његова улога је више усмерена ка анализи корисничког захтева које производ треба да задовољи.

[4] Објекат dslLanguage:DSL

Графички језици представљају први ниво у архитектури решења за развој и производњу софтвера помоћу наменских графичких језика. У првом моделу

производње софтвера доменски језик (ДСЛ) се не користи, односно није обавезан део производног процеса.

[5] Објекат modToolFeatures:ObjModifier



Објекат типа модификатор илуструје намеру и начин модификације постојећих модела производње софтвера и алате који се за то користе. Треба унапредити сваки од четири нивоа у архитектури решења који користе графичке језике за моделовање. Модификација може да буде додавање нових својстава, допуњавање постојећих и избацивање неких својстава или функција. Модификатор је на овом моделу декомпонован преко релације ■5-D-■4, ради детаљнијег приказа архитектуре ДСМ.

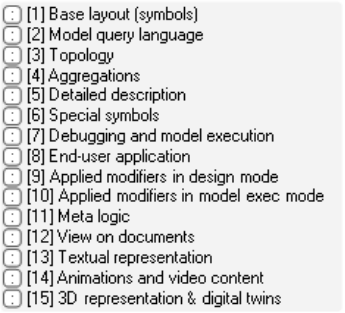
[6] Објекат DsmTool:ToolForMDSD

Овај објекат представља алат за моделовање помоћу наменских графичких језика. У сарадњи са експертима, коришћење овог алата нам омогућава ефикаснији развој и производњу унутар једног циклуса. Ови алати не користе инверзни инжењеринг, већ све иде од модела према готовом производу.

[7] Објекат codeGen:Generators

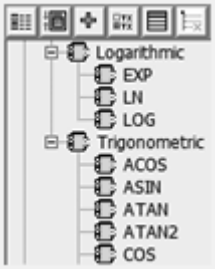
Генератори кода су посебан ниво у архитектури решења са наменским графичким језицима и служе да се модели преводе у изворни код, конфигурационе фајлове, документацију, корисничке апликације и разне друге творевине. Они допуштају извесну слободу у интерпретацији модела.

[8] Објекат ModelView:ObjModifier



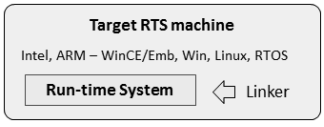
Овај објекат, по типу модификатор као и (■5), приказује скуп потребних погледа на модел или варијација изгледа (графичке синтаксе) за продуктиван развој и примену језика за моделовање и уопште за потребе организовања ефикасног производног процеса. Алати за моделовање најпре треба да подрже једноставне графичке представе језичких концепата. Моделе треба претраживати помоћу наменских упитних језика. Поглед на тополошка својства модела служи да нас усмери на елементе и везе без детаља, нпр. на механичка својства робота. Агрегацијама и супституцијама групишемо и замењујемо неке делове модела да бисмо усмерили пажњу на друге делове. Детаљан опис нам треба за документацију, презентације и филмове. Специјални симболи из фонтова су најбржи начин да се направе прве графичке представе језичких концепата. Стандардни дибагери из програмерских алата нису намењени за дибагирање модела. Из модела се генеришу корисничке апликације. Потребан нам је бар по један поглед на сваку од понуђених апликација. Спецификација, интерпретација и тестирање варијација модела су посебно сложени послови, па све то треба пребацити на ниво графичког интерфејса и модела, тако да се помоћу генератора кода утиче на начин интерпретације. Мета-логика и мета-аритметика су важни за анализу грешака у генерисаном коду, ради испуњавања захтевних стандарда и ради сертификације производа. Поглед на документе реализује се преко генератора кода, који генеришу документацију. Текстуална репрезентација модела је важна ради размене са другим развојним платформама и архивирања. Из модела се генеришу презентације и видео садржаји. Представљање у 3Д простору је посебно важно код језика за моделовање у роботици, па је томе посвећена велика пажња.

[9] Објекат tarFramework:Framework



Одредишно окружење (енг. framework) служи да у моделу производње прикажемо које све постојеће библиотеке и сервисе, односно како и коју инфраструктуру користимо. Генератори кода улазне моделе преводе у изворни код и позиве функција библиотека и сервиса. Преко њих генератори кода, као и апликације које они генеришу, комуницирају са инфраструктуром. Исти модели се могу извршавати на различитим инфраструктурама, коришћењем различитих окружења.

[10] Објекат tarSystem:TargetSystem



У роботици одредишни систем може да буде оперативни систем, наменски роботски систем или систем за аутоматизацију. У контексту модела производње, одредишни системи могу бити разни системи за вођење верзија производа, архиве, као и било која инфраструктура за континуирану интеграцију и дистрибуцију нових верзија софтвера.

[11] Објекат UmlTool:ToolForMDSD

Овај објекат представља алате за моделовање, као што су УМЛ алати, али оне који не користе наменске графичке језике. Нису део архитектуре ДСМ-а. УМЛ алати се користе углавном у појединим фазама према првом моделу производње софтвера.

[12,13,14] Објекти Analysis, Design, Development:StepInSDLC

Представљају фазе анализе захтева, дизајнирања и развоја софтвера.

[15,16,17] Објекти Testing, Deploymenet, Maintenence:StepInSDLC

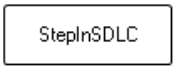
Представљају фазе тестирања, увођења и одржавања софтвера.

Модели развоја и производње софтвера садрже елементе различитих типова објеката, релација и улога, са припадајућим својствима (енг. property). Типови елемената и правила повезивања дефинишу тип модела, односно мета-модел или језик за моделовање.

Циљ овог првог примера није детаљна спецификација неког језика, већ је он увод у начин како су касније у књизи представљени примери модела и дефиниције језика преко мета-модела. Сви различити типови објеката груписани су у моделе који су приказани на слици 2. Модели који описују дефиницију језика за моделовање се називају мета-модели језика и налазе се на једном степену апстракције више него модел на слици 1. Ради јаснијег приказа направљене су две одвојене целине, два подмодела. У горњем подмоделу су они типови објеката који могу да се везују преко релације типа Flow, а у доњем су они који могу де се везују преко релације типа Rel. Релације Flow служе за приказ односа између елемената у којима неки елементи модела производе резултате који се даље користи у некој од следећих фаза или корака производње софтвера. Релације типа Rel служе да се изразе различите врсте груписања и односа између алата и корака у развоју и производњи. Осим ова два типа, на почетном моделу имамо и релације типа M (ObjMod), односно модификације. Ове релације служе за повезивање елемената модела са листом њихових модификација, које могу да постоје или које треба урадити да би се унапредили развој и производња. Сваки тип објекта може бити обухваћен бинарном релацијом типа <M>, али тако да је са

друге стране објекат типа модификатор. Зато нису посебно приказани на слици 2. У првом примеру на слици 1 релације типа Flow су приказане на два начина: као кругови унутар којих је слово F, и као кругови са по два усмерена кружна исечка.

[1] Објекат StepInSDLC_DVMExMetaObj:StepInSDLC

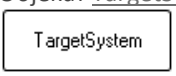


Овај тип објекта служи за представљање фаза или корака у производњи софтвера. Приказан је у облику заобљеног правоугаоника са текстом у средини, који има назив тог типа, као и сви други објекти на мета-моделу.

[2] Објекат SoftProdDevModel_DVMExMetaObj:SoftProdDevModel

Овај тип објекта служи за представљање модела производње или развоја софтвера.

[3] Објекат TargetSystem_DVMExMetaObj:TargetSystem

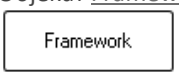


Служи за приказ различитих одредишних система за које се генерише или производи софтвер.

[4] Објекат DSL_DVMExMetaObj:DSL

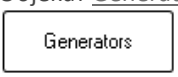
Служи за приказ различитих врста наменских графичких језика за развој и производњу софтвера.

[5] Објекат Framework_DVMExMetaObj:Framework



Служи за приказ библиотека и сервиса који се користе при генерисању кода клијентских и било којих других апликација.

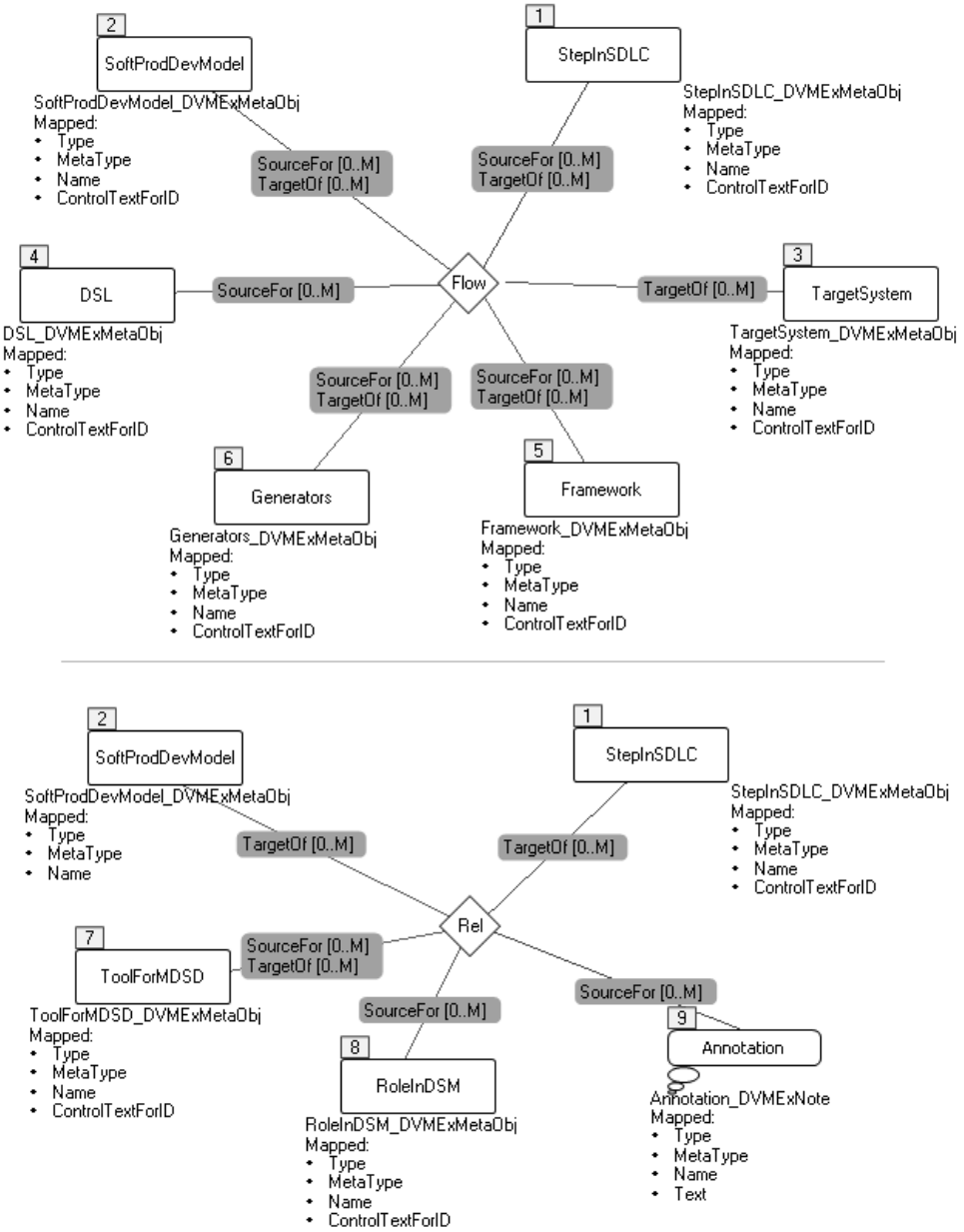
[6] Објекат Generators_DVMExMetaObj:Generators



Служи за приказ различитих типова генератора кода који моделе преводе у софтвер и друге творевине. Генератори кода су углавном текстуални језици, али се из мета-модела графичких језика неки генератори могу генерисати аутоматски. То у пракси обично није потребно, јер језици генератора кода имају једноставну синтаксу.

Мета-модел, осим објеката и релација, има улоге и својства модела и свих мета-типова. Када је неки тип објекта на мета-моделу повезан са релацијом преко улога, чија имена се овде налазе на средини линије, онда то значи да је

дозвољено повезивање тог типа објекта за те релације коришћењем приказане улоге, са кардиналитетима који су задати унутар угластих заграда. Веза .Generators~SoureFor[0..M]>Flow значи да у моделима који приказују ток производње, или развоја софтвера за исту релацију, можемо везати нула или више генератора кода.

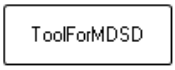


Слика 2: Мета-модел језика за опис типова развоја и производње софтвера

То значи да их негде уопште не користимо, а да негде имамо једну или више група генератора или техника за генерисање кода. Генератор може бити и извор и корисник неког садржаја, ка свим и од свих типова објеката, што је приказано везама .Generators~SoureFor[0..M]>Flow и .Generators~TargetOf[0..M] >Flow.

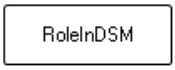
Објекти који чине други подмодел, а који су приказани у доњем делу слике, делом су исти као и они који се везују преко релација типа Flow (1,2), али имамо и три нова. То су алати за моделовање (7), улоге које остварују учесници у развоју и производњи софтвера (8) и разне напомене (9). Веза .RoleInDSM~SoureFor[0..M] >Rel показује да се нека улога може остваривати у више корака развоја и производње, са различитим алатима, у различитим производним моделима. Значење преосталих објеката из другог подмодела је следећа:

[7] Објекат ToolForMDSD_DVMExMetaObj:ToolForMDSD



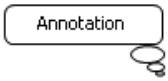
Алат за моделовање.

[8] Објекат RoleInDSM_DVMExMetaObj:RoleInDSM



Улога у процесу развоја или производње софтвера.

[9] Објекат Annotation_DVMExNote:Annotation



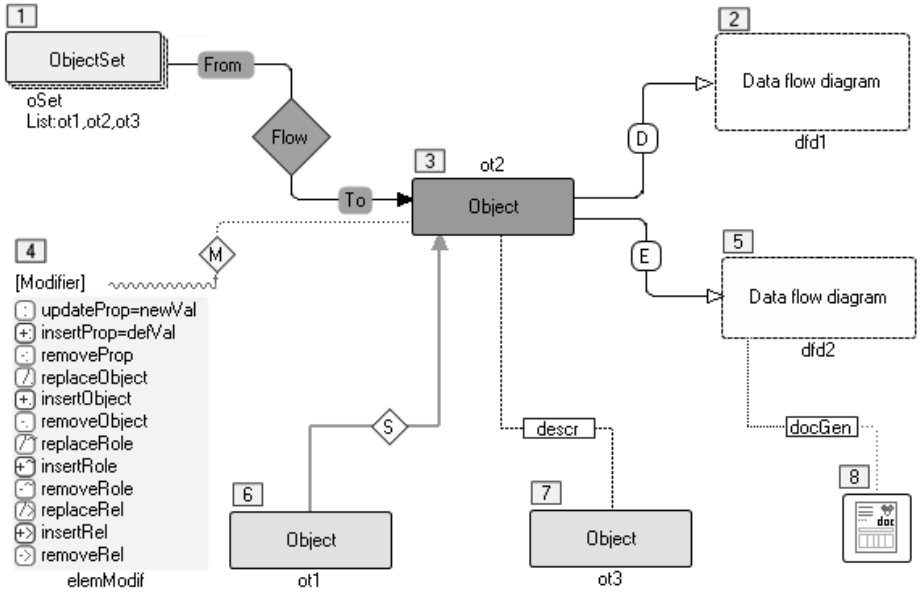
Напомене које се односе глобално на модел, групу елемената или на појединачне елементе модела.

Приликом описивања дозвољених веза типова елемената преко типова улога и типова релација користи се облик који има симболе „.“, „~“ и „>“. Ови симболи се користе за идентификацију мета-типа или мета-концепта. Тачка служи за објекте, таласаста цртица за улоге, а знак „веће од“ за релације. Осим њих, за својства се користи знак „:“, за портове „#“, а за моделе „*“.

Ако преко модела хоћемо да прикажемо типове концепата који се користе у графичким језицима за моделовање, правила њиховог повезивања и ограничења, добијамо модел који се зове мета-мета-модел, као што је приказано на слици 3. Овај мета-мета модел приказује да сви елементи језика припадају објектима (■1), дијаграмима тока (■2,5), улогама (From,To) и релацијама (Flow,D,E,S, descr, docGen) које се односе на ток (■2), подтипове (■6), описе (■7), модификације (■4), декомпозиције (■2) и тзв. експлозије (■5) модела. Исте улоге може да остварује један или скуп објеката, али не и улогу подтипа, описа и модификације. То значи да извођење подтипа увек везујемо за један надтип, да се опис односи на један објекат модела и да се модификација такође односи на један објекат модела. Полазећи од овако једноставних приказа, приликом конструкције језика и њихових представа, уводимо сложеније типове елемената и репрезентације

које су у складу са реалним потребама у области примене језика. Док је релација декомпозиције углавном јасна и представља детаљан приказ неког објекта преко додатног модела (релација по висини), експлозија се више користи за приказ елемената модела који су углавном на истом нивоу апстракције, као и модел (релација по ширини).

На самом почетку конструкције језика за моделовање у роботици треба да имамо у виду да ће успешност примене робота на решавање одређених проблема зависити и од експертског знања у роботици (механици, електроници и софтверу), а не само од експертског знања стручњака у нпр. некој биолабораторији у којој се нешто роботизује. Успех ће, дакле, зависити и од познавања својстава механике и алата који изводе операције, од својстава електронике која покреће механику и комуницира са софтвером, од својстава окружења у ком се изводе операције и од поштовања захтевних стандарда за поуздан и безбедан рад, било робота самостално, било у колаборацији са човеком. То нам наговештава да ће за решавање проблема морати да се конструише више наменских језика који су појединачно усмерени на експертско знање свега претходно набројаног. Ти језици мора да буду интегрисани у једну функционалну целину из које је могуће добити све потребне творевине за поуздану управљачку логику. Како свака од компоненти која чини робот може имати варијације, нпр. варијације тополошких својстава роботске руке или протокола за комуникацију са контролером, то и језици за моделовање мора да подрже једноставан опис тих варијација.



Слика 3: Мета-концепти и њихов приказ (мета-мета модел)

Када цртамо моделе да бисмо описали нека од својстава робота, тада користимо само неке типове елемената. Због тога моделе групишемо у типове модела. Тип модела је, дакле, одређен скупом типова језичких концепата који се користе за моделовање неких својстава робота или неког типа посла који робот обавља. У роботској пракси се разликују типови модела који служе за опис:

- тополошких својстава роботске руке и алата,
- кретања и операција у простору,
- окружења у ком се обављају операције,
- управљачке логике и протокола контролера,
- управљачке табле оператера и алата,
- поступка калибрације и
- правила којима се испуњавају стандарди безбедног рада.

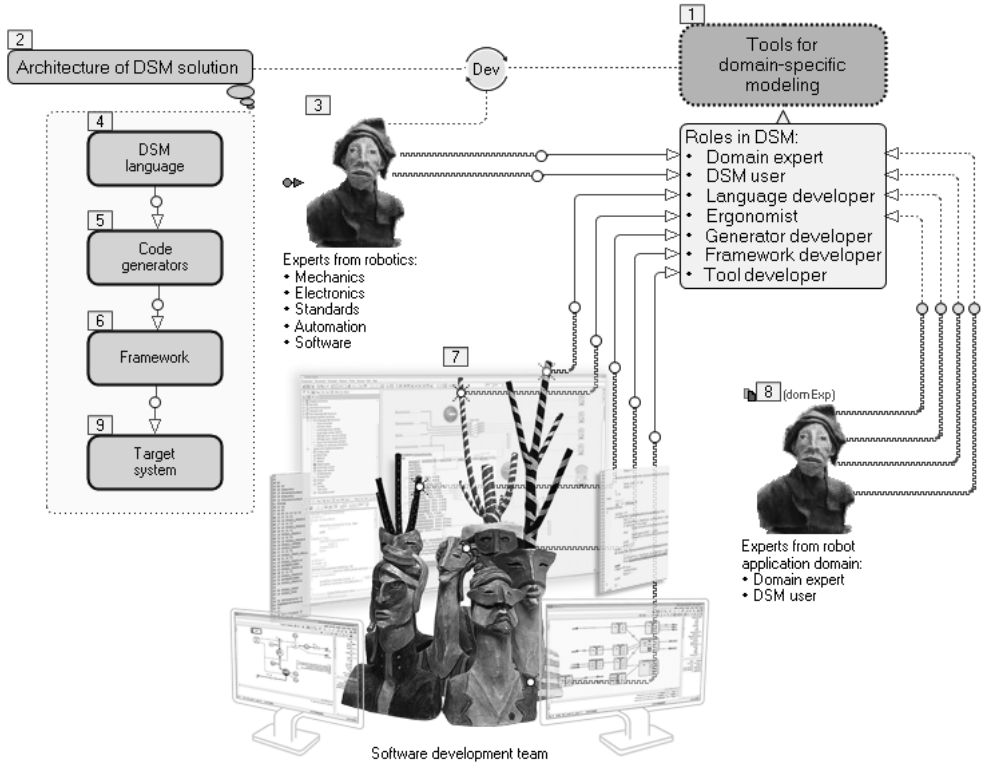
Следећих седам поглавља о наменским графичким језицима за моделовање у роботици посвећено је конструкцији и примени језика за побројане типове модела. Та поглавља се односе на први ниво у архитектури решења за наменско моделовање, на ниво наменских графичких језика. Конструкција језика илустрована је примерима који су добро проверени у пракси. Добри примери су основа да формирамо један нов поглед на методологију развоја софтвера за роботику и аутоматику. Ти примери су присутни и у каснијим поглављима, која описују преостале нивое архитектуре решења за наменско моделовање у роботици: генераторе кода, окружење и одредишне системе који извршавају спецификације.

Садржај ове књиге је подељен на четири целине, од којих свака одговара једном нивоу у архитектури решења за развој и производњу софтвера помоћу наменских графичких језика. Највећи број поглавља односи се на језике за моделовање. Ради илустрације практичне користи од овакве архитектуре, књига садржи и поглавља о генераторима кода, окружењу и одредишном систему за роботику и аутоматику.

1.1 Улоге учесника у развоју робота и софтверских алата помоћу наменских графичких језика

Успех у сваком послу, па и у развоју робота помоћу наменских језика и алата, зависи од многих предуслова. Свакако најбитнији предуслов је да посао ради креативан и искусан развојним тим, који се састоји од стручњака потребних профила. Њима се поверавају улоге које они најбоље могу да обаве и за које су стручни. Проблем аутоматизоване производње софтвера за роботе се не разликује од развоја софтверских алата који служе за конструкцију наменских језика за било коју област примене. Због тога се питања и проблеми који су описани у овој књизи не своде на приказ једног или групе језика за моделовање у роботици, већ на софтверско инжењерство уопште. Послови у решавању проблема могу да се групишу у две улоге. Прва је конструкција наменских језика, генератора кода, окружења и одредишног система који извршава моделе, а друга је развој софтверских алата помоћу којих се све претходно побројане компоненте развијају и проверавају систематично. Као што интерпретација модела, нпр. управљачких процеса, зависи од ваљаности језика за моделовање, од својстава језика генератора кода, окружења (библиотека и сервиса) и одредишног система, тако и интерпретација операција и функција алата за моделовање, зависи од свих нивоа у архитектури софтверског алата. Ако генераторе кода проширимо тако да

на исти начин приступамо и оперишемо са логичким и имплементационим својствима, онда су тзв. платформски независни и платформски зависни модели готово исто, па престаје потреба да се они и методолошки одвајају. Логички концепти се пресликавају (енг. mapping) у имплементационе, што гарантује добре понуђене апликације без икаквих додатних трансформација и било каквих међукорака. Ако направимо добро окружење за развој алата, решили смо још већи проблем, а то је развој алата који се ослања на добре функције. На крају, ако имамо добар одредишни систем за брзу проверу алата и наменских језика, успех је само питање времена. Референтна имплементација графичких језика и визуелно дибагирање су најпоузданији ниво тестирања језика и алата. Такав поступак тестирања се назива извршавање модела. Повратна веза од извршиоца модела нам обезбеђује информације за профињавање модела који се извршавају и језика помоћу којих су ти модели задати.



Слика 4: Улоге у развоју језика и алата

На слици 4 приказана је архитектура решења које се заснива на доменском моделовању[1], односно на конструкцији и употреби наменских графичких језика у роботици (■2). Ови језици служе за цртање модела, који се помоћу генератора

[1] Архитектура приказаних решења за развој софтвера у роботици и аутоматици заснива се на општој архитектури ДСМ-а, која је детаљно описана у књизи "Domain-Specific Modeling: Enabling Full Code Generation", чији аутори су Стивен Кели (Steven Kelly) и Јуха-Пека Толванен (Juha-Pekka Tolvanen). Бројни стручни радови из области наменских графичких језика доступни су на www.dsmforum.org.

кода (■5) преводе у извршни код и друге творевине, користећи при томе разна окружења (■6) и одредишне системе (■9). У овом приступу нема инверзног инжењерства зато што није потребно. У каснијим поглављима приказаћемо неколико наменских језика (■4) и функције развојних алата за ДСМ (■1). Експерте који су нам потребни за овај посао поделили смо у две групе: на експерте из области конструкције робота (■3) и на експерте из области примене робота (■8). И једни и други су подједнако важни и као експерти из својих области и као први корисници свега што правимо. Њихов сусрет и рад организује развојни ДСМ тим (■7) оног тренутка када настану први кориснички захтеви са првим елементима језика струке. Тада и они постају део тима.

У оваквој организацији, људи који треба да створе решење налазе се у групи ■7. Сваки од њих има по неку од улога која је усмерена на креирање језика, ергономска својства алата, генераторе кода и докумената, развој окружења и развој алата. Исту улогу може да обавља више чланова групе ■7, али и обрнуто, један члан групе може да обавља више улога, у зависности од сложености и обима посла.

Методологије развоја софтвера недовољно одвајају улоге експерата из домена примене робота (■8) од оних који су експерти за роботику (■3). Експерти из групе 3 добијају и улоге из групе ■8, иако су им професије различите. Такав неадекватан начин додељивања улога и организације развојног тима на слици је приказан тако што су експерти из групе ■8 „мека копија“ (енг. shallow copy) експерата из групе ■3. У том случају се групе привидно разликују, али им је функција иста. Последица такве организације је да наменски језици за моделовање неће бити добар алат за њихове будуће кориснике. Зато експерти из групе ■8 не треба да буду „мека копија“ експерата из групе ■3. Ова књига је пре свега намењена експертима из групе ■3 и софтверским развојним тимовима (■7). Она садржи основе методологије и значајна сажета искуства из праксе. Случајним читаоцима ове књиге који припадају групи ■8 она може да помогне у грађењу језика струке.