
Пловдивски университет
„Паисий Хилендарски“

Факултет по математика и информатика

Катедра „Компютърни системи“


# Отворени, интерактивни и динамични визуални езици за програмиране

Автореферат

на дисертационен труд
за присъждане на образователна и научна степен “доктор“
в област на висше образование 4. Природни науки, математика и информатика;
Професионално направление 4.6. Информатика и компютърни науки;
докторска програма Информатика


*Редовен докторант:*
Васил Георгиев Василев

*Научни ръководители:*
проф. д-р Станимир Стоянов,
гл. ас. д-р Александър Пенев



Пловдив, 11 Август 2015

Дисертационният труд е обсъден и насочен за защита от катедрения съвет на Катедра „Компютърни системи“ на Факултет по математика и информатика при Пловдивски университет „Паисий Хилендарски“ на 17.06.2015 г.

Дисертационният труд Отворени, интерактивни и динамични визуални езици за програмиране съдържа 163 страници. Библиографията включва 114 източника. Списъкът на авторските публикации се състои от 5 заглавия. В автореферата номерацията на формулите, цитиранията и примерите съвпада с тяхната номерация в дисертационния труд.

Защитата на дисертационния труд ще се състои на 18.09.2015 г. от 11 часа в Заседателната зала на Нова сграда на Пловдивски университет „Паисий Хилендарски“ (бул. “България” No 236). Материалите по защитата са на разположение на интересуващите се в секретариата на Факултет по математика и информатика на Пловдивски университет „Паисий Хилендарски“, (бул. “България” No 236, Пловдив), каб. 330, всеки работен ден от 8:30 до 17:00 ч.

Автор:
**Васил Георгиев Василев**

Заглавие:
**Отворени, интерактивни и динамични визуални езици за програмиране**

Рецензенти:
Проф. д-р Минчо Петков Сандалски, ФИСН при ПУ “Паисий Хилендарски”
Проф. д.т.н. инж. Тодор Атанасов Стоилов, ИИКТ на БАН

Университетско издателство “Паисий Хилендарски”
Пловдив, 2015 г.

Тираж 100 бр.

# Съдържание

Глава 1

# Обща характеристика на дисертационния труд

Настоящият дисертационен труд е посветен на архитектурата, дизайна и етапите на изграждане на динамична, интерактивна и отворена метасистема, която улеснява създаването на визуални езици за програмиране. Разглежда се и прототипна реализация на система, илюстрираща основните принципи и представяща практическа програмна реализация на настоящата разработка.

Обект на изследването на този дисертационен труд са отворените, интерактивни и динамични визуални езици за програмиране (ВЕП) и визуалните системи, поддържащи ги. Предмет на изследване са градивните елементи на отворените, интерактивни и динамични ВЕП (ОИДВЕП) и тяхното приложение и евентуално обобщение. В изследването се формулира и доказва следната хипотеза: възможен е общ подход за проектиране и реализация на набор от инструменти в метаинструментариум, който позволява да се отдели общата инфраструктура при изграждане на ОИДВЕП. С помощта на метаинструментариума могат да се изследват различни аспекти на ОИДВЕП. Използването на този подход има за цел да ускори бързината на разработката на такъв тип ВЕП, като същевременно и намали сложността на цялостния процес. Като емпирично доказателство за валидността на хипотезата е конструиран прототипен метаинструментариум (SolidV), базиран на предложения модел. Неговата преизползваемост и приложимост ще бъде по-детайлно анализирана посредством изграждане на различни визуални системи (ВС).

## 1.1 Актуалност на тематиката

Съвременните изчислителни машини ни предоставят почти напълно виртуализирана комуникационна среда. Нейните свойства и характеристики отдавна задминават конвенционалните среди като хартията, например. Телекомуникационните и електронните медии целят да виртуализират комуникационната среда и да я направят

по-достъпна за потребителите. Обикновено целта е да се предостави среда, която наподобява хартия или в най-добрия случай електронна медия като радио или телевизия. Те обаче по дефиниция са по-статични, по-неинтерактивни и по-затворени, поради принципите им на функциониране.

Въвеждането на интерактивен графичен интерфейс променя качествено средата за комуникация, включваща машини. Тази среда е много по-динамична, много по-интерактивна и много по-отворена за разширения. Например рисуването на линия върху лист хартия има малко общо с рисуването на линия върху таблет от гледна точка на динамика и интерактивност. Рисуването в компютърна комуникационна среда представлява много повече от оставяне на въглеродна следа на парче хартия [1]. Създава се модел на изображението, визуализация, може да се прилагат математически трансформации, може да се преизползват нарисуваните елементи и т.н.

В същността си, разработката на програми за компютърната комуникационна среда представлява сложна комуникация между човек и компютър. Основните ѝ принципи са от времето на перфо-картите, т.е. тя е предимно базирана комуникация чрез текст върху симулирани "листове" хартия, използвайки контекстно-независими езици – езици за програмиране (ЕП).

От друга страна, визуалните изображения са продуктивна метафора за мислене. Те са конкретни и лесно управляеми, като имат добри способности за представяне. ЕП могат да бъдат много повече от пасивна среда за комуникация, те могат да увеличат способностите на потребителите да учат и мислят. "Креативното мислене включва сечението на два обикновено отделни мисловни контекста, използвайки изображения и от двата" [5]. Бъдещите ЕП биха могли да се използват по-ефективно от компютърната комуникационна среда и едновременно да предоставят по-добри инструменти за комуникация между разработчик и компютърна система (КС).

Неформално, работата изследва възможността за предоставяне на по-активна среда за комуникация при процеса на програмиране. Опитва се да предостави инструменти, които биха улеснили използването на визуални метафори по време на програмиране, мислене и обучение.

## 1.2 Цели и задачи на дисертационния труд

**Основна цел на дисертационния труд е да се създаде модел и прототип на метаинструментариум, улесняващ изграждането на отворени, интерактивни и динамични визуални езици за програмиране.**

За постигането на това са дефинирани следните подцели и задачи:

**Ц1** Да се създаде теоретичен модел и архитектура на метаинструментариум за създаване на ОИДВЕП. Задачи:

*Ц1.31* Създаване на теоретичен модел за метаинструментариума;

*Ц1.32* Създаване на архитектура за метаинструментариума.

**Ц2** Да се реализира прототип на метаинструментариум, отговарящ на дефинираната архитектура за създаване на ОИДВЕП. Задачи:

*Ц2.31* Разделяне на прототипа на компоненти;

*Ц2.32* Реализация на основните компоненти.

**Ц3** Да се разработят ВЕП/ВСП, основани на концептуалния модел. Да се изследва и покаже преизползваемостта на метаинструментариума и неговите компоненти. Задачи:

*Ц3.31* Да се реализират ОИДВЕП, използвайки метаинструментариума;

*Ц3.32* Да се анализира преизползваемостта на компонентите на метаинструментариума.

**Реализационни цели на работата:**

- **Общоприложимост** – основополагащите концепции и принципи трябва да са приложими върху широк набор от приложни области;
- **Отвореност** – способността на метаинструментариума да бъде разширяван в едно или повече направления;
- **Разширимост** – способността да бъде добавяна нова функционалност чрез модули;
- **Гъвкавост** – лесното адаптиране към различни приложни области;
- **Слаба свързаност и модулност** – слабата свързаност повишава възможността даден реализационен елемент да бъде използван самостоятелно и да позволява дадената система да бъде изследвана, променяна и разширявана. Модулността е реализационният аспект на слабата свързаност;
- **Платформена независимост** – метаинструментариумът трябва да поддържа максимално количество платформи.

## 1.3 Структура на дисертационния труд

Настоящият дисертационен труд е организиран в няколко глави както следва:

- Глава 1, "Увод", въвежда в проблемната област, като проследява развитието на КС и ЕП през годините.
- Глава 2, "Обзор на визуални езици и среди за програмиране", проследява развитието на проблемата област, като разглежда по-подробно различни изследователски и практически достижения. Предлага техен сравнителен анализ.
- Глава 3, "Модел за създаване на отворени, интерактивни и динамични визуални езици за програмиране. Метаинструментариум", навлиза в дълбочина и ширина на основополагащите концепции за създаването на теоретичен модел и архитектура на метаинструментариум за създаване на отворени, интерактивни, динамични визуални ЕП. Прави теоретична обосновка на избраните концептуални и архитектурни решения.
- Глава 4, "Архитектура и прототипна реализация на метаинструментариума", дискутира покомпонентната реализация на прототип на метаинструментариум за създаване на отворени, интерактивни, динамични визуални ЕП. Излага практически аргументи за конкретния подбор на архитектурни решения и разглежда техническите особености на компонентите на метаинструментариума.
- Глава 5, "Приложения и резултати", предлага някои (известни на автора) приложения на метаинструментариума при създаване на визуални ЕП и приложения. Разглежда накратко някои реализационни аспекти.
- Глава 6, "Заключение", формулира цялостните приноси и резултати от работата. Изброява направените от автора постижения в областта на дисертацията и обучението на студенти.

Глава 2

# Кратко съдържание на дисертационния труд

## 2.1 Глава 2 – Обзор на визуални езици и среди за програмиране

*Настоящата глава развива мотивацията на дисертационния труд по-детайлно. Проследява развитието на областта, чрез изброяване на важните еволюционни стъпки както в КС, така и в ЕП. Дефинира фундаментални за работата понятия, като "програма", "интерактивност", "визуално програмиране" и "програмна визуализация". Дефинира понятието "визуална среда за програмиране" в термините на визуален език за програмиране. Прави обстоен преглед и сравнение на съществуващи системи.*

### 2.1.1 Основни понятия

В областта има няколко сравнително общоприети дефиниции на основни понятия. Настоящата работа използва дефинициите на Myers [6]:

- Програма – "множество оператори, които могат да бъдат разглеждани като модул от гледна точка на КС и използвани за насочване на нейното поведение." [7].
- Визуално програмиране – се отнася до всяка система, която позволява на потребителя да дефинира програма в две (или повече) измерения. Конвенционалните текстови езици не се възприемат като двумерни, тъй като компилаторът или интерпретаторът ги обработва като един едномерен низ. Визуалното програмиране включва конвенционални блок схеми (flow charts) и графични езици за програмиране. То не включва системи, които използват конвенционални (линейни) ЕП да дефинират изображения.

- Визуален език за програмиране е език за програмиране, съдържащ конструкции, които са или по същество (inherently) или с наложено (superimposed) многомерно представяне" [6].
- Визуална среда за програмиране е интегриран набор от инструменти, които поддържат редактиране, анализ и изпълнение на визуални програми" [8].

### 2.1.2 Визуални среди и езици за програмиране. Метаинструменти

Визуалните среди за програмиране (ВСП) и ВЕП еволюират през годините, като формират техни собствени домейни [9]. Исторически, ВСП и ВЕП са взаимосвързани – например концепции, появили се първоначално в едните, бързо биват приети от другите. На практика, ВСП могат да се разглеждат, като домейн-специфичен, отворен, динамичен и интерактивен ВЕП. Някои ВСП могат да реализират повече от един ВЕП. Например типична ВСП реализира домейн-специфичен език за дизайн на графични потребителски интерфейси и друг език за моделиране с UML.

В настоящата работа ВЕП и ВСП се използват като синоними. Терминът ВС се използва за системи, реализиращи ВЕП, ВСП или такива с елементите на програмна визуализация. Работата условно разделя разгледаните ВС на 4 поколения (I – 1966 до 1980, II – 1981 до 1990, III – 1991 до 2000, IV – 2001 до 2015), в зависимост от използваните технологични средства за тяхната реализация. Липсата на достатъчно изчислителна мощ, теоретични познания и концепции за абстракция представляват естествени ограничители на функционалността на разработените ВЕП. Забелязва се, че ограниченията се променят практически на около 10 години.

С поколенията ВЕП се превръщат в мултидисциплинарно направление, комбиниращо редица подобласти на информатиката и информационните технологии. Създаването на подобен род визуални системи с общо предназначение става изключително сложно и това е една от причините домейн-специфичните ВЕП да процъфтяват. Ясни примери за наложили се домейн-специфични ВЕП са графичните дизайнери за създаване на графични потребителски интерфейси (ГПИ) и UML.

Появяват се също и т.нар. метаинструментариуми, които предлагат блокове за изграждане на инструментариуми за ВЕП, намалявайки степента на сложност. Това също спомага за разширяване на обхвата на получените инструментариуми за ВСП. Включват се метаинструменти, разработват се метасистеми и др. Част от причините за появата на метаинструментариумите е, че различните ВЕП се развиват само в някои направления. Например много малко от тях са едновременно отворени за

разширения, интерактивни и динамични. Реализацията на тази функционалност, без специализирани инструменти, е много трудна.

### 2.1.3 Сравнителен анализ

Направеният сравнителен анализ в настоящата работа има за цел да покрие подробно разгледаните в работата системи. Голяма трудност представлява намирането на общи критерии и сравняването на системи, съществували преди 40 години и настоящи такива. Още повече, че достъпът до кода на системите понякога отсъства (прекалено са стари или са за комерсиално използване). При много старите системи, платформите, на които се е изпълнявал, не са налични.

Сравнителната характеристика има за цел да обърне внимание на определени страни на разгледаните ВЕП и да оцени (понякога субективно) качествено характеристики като: отвореност; динамичност; интерактивност; платформена независимост; лиценз.

| ВС | Пок. | Отвореност | Динамичност | Интерактивност | Платформена независимост | Лиценз |
|---|---|---|---|---|---|---|
| AMBIT | I | Липсва | Ниска | Ниска | Липсва | ? |
| GRAIL | I | Ниска | Ниска | Ниска | Липсва | ? |
| Pygmalion | I | Липсва | Ниска | Ниска | Липсва | ? |
| Prograph | II | Ниска | Добра | Много добра | Unix, OSX | Комерс. |
| IGIP | II | Добра | Добра | Добра | ? | ? |
| VisSim | II | Много добра | Много Добра | Много Добра | Win | Комерс. |
| Chimera | II | Липсва | Ниска | Ниска | ? | ? |
| DRAKON | III | Липсва | Много Добра | Липсва | Win, Unix, OSX | Свободен |
| AGG | III | Ниска | Много Добра | Добра | Win, Unix, OSX | Свободен |
| PROGRES | III | Добра | Добра | Много добра | Solaris, Unix | Свободен |
| Mashups | IV | Липсва | Много добра | Добра | Win, Unix, OSX | Свободен |
| DataSlave | IV | Липсва | Много добра | Добра | Win | Комерс. |
| OpenMusic | IV | Липсва | Много добра | Липсва | Win, Unix, OSX | Свободен |
| KTechLab | IV | Ниска | Добра | Липсва | Unix | Свободен |
| OpenWire | IV | Ниска | Много добра | Добра | Win, Android | Свободен |
| Escalante | - | Много добра | Много добра | Добра | Unix | Свободен |
| DiaGen | - | Много добра | Добра | Добра | Win, Unix, OSX | Свободен |
| IPSEN | - | Много добра | Добра | Липсва | Unix | Свободен |
| RSA | - | Добра | Добра | Ниска | Win, Unix, OSX | Комерс. |
| GME | - | Липсва | Добра | Липсва | Win | Свободен |
| MetaEdit | - | Липсва | Много добра | Много добра | Win, Unix, OSX | Комерс. |
| EclipseGMF | - | Много добра | Добра | Ниска | Win, Unix, OSX | Свободен |

Таблица 2.1: Сравнение на някои характеристики на разгледаните ВЕП.

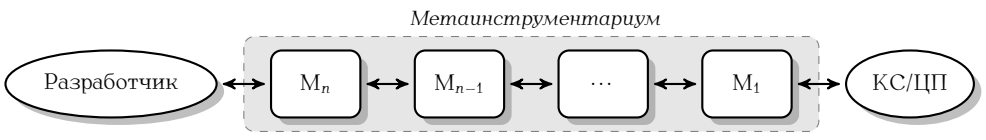
За сравнение на бързодействие е трудно да се говори, поради разнородността на разгледаните системи. На практика е невъзможно да се конструира единен тестов пакет с тестове за производителност на посочените системи.

От анализите дотук става ясно, че реализацията на ВЕП е много трудоемко начинание. Богатите на информация представяния при ВЕП поставят предизвикателства пред разработчиците от много различни подобласти като визуализация, работа с прозорци, ГПИ и транслация. Първичният източник на проблеми при създаване на ВЕП е сложността на мултидисциплинарното естество на задачата. На практика, малка част от времето се отделя за формулирането на специфичните за ВЕП функции, а другата част остава за построяване на неспецифичната за езика инфраструктура.

## 2.2 Глава 3 – Модел за създаване на отворени, интерактивни и динамични визуални езици за програмиране. Метаинструментариум

*Настоящата глава се стреми да даде солидна теоретична основа, целяща да подпомогне и улесни реализацията. Дефинира характеристики на ВЕП като отвореност, интерактивност и динамичност. Разглежда компютърната програма като съвкупност от модели, които са междинни стъпки при транслация. Обсъжда дейности върху моделите като междумоделни трансформации, еволюция и оптимизация на модели. Обяснява значението на визуалните модели и взаимодействията между тях. Дискутира реализационни техники за контрол на сложността при създаване на сложни системи. Аргументира използването на архитектура, базирана на компоненти.*

За улесняване на работата в конкретната област е изключително полезно, неспецифичната за ВЕП инфраструктура се обособи в отделна подсистема, която позволява да се намали сложността на работа в областта. Предложеният теоретичен модел на фигура 2.1 отделя общите части, разтоварвайки разработчиците на ВЕП в някои аспекти. Предвид разнородните цели и дейности на подсистемата е логично тяхното изолиране в модули ($M_n$).




Фигура 2.1: Концептуална схема на метаинструментариум.

Описаната подсистема може да бъде наивно класифицирана като транслатор за ВЕП, защото тя осигурява комуникацията между човек и КС посредством език за програмиране. Тя е медиатор на взаимодействието между човек и КС, като основният

фокус е върху динамичната, интерактивната и отворената комуникация. За разлика от транслаторите, върху метаинструментариума могат да се изграждат ВЕП, но могат да се изграждат и специализирани инструментариуми за разработка не само на ВЕП, но и други инструменти. Тези характеристики класифицират транслатора като метаинструментариум. Той моделира взаимодействие между човек и КС чрез реализация на преизползваеми компоненти и класове функционалност и в двете посоки на комуникация:

- Права посока – човек-машина – въвеждане на входните данни и понижаване на абстракцията.
- Обратна посока – машина-човек – извеждане на изходните данни и повишаване на абстракцията.

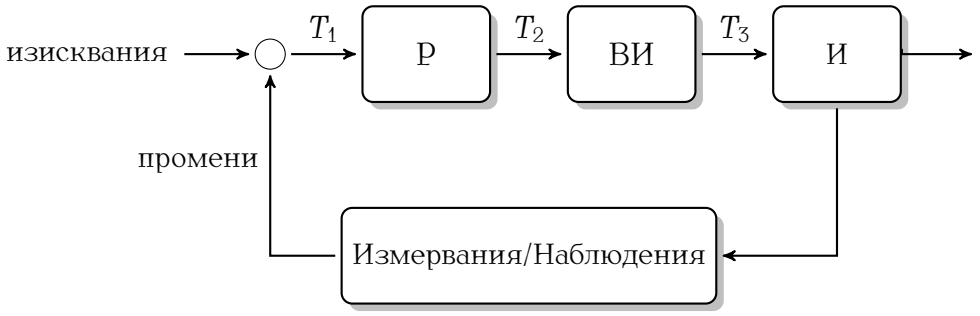
### 2.2.1 Отвореност, интерактивност, динамичност

За да се ограничи двусмислието, настоящата работа дефинира отвореността като комбинация от следните критерии:

- Разширимост – по какъв начин е възможно ВЕП да бъде разширяван. Може ли в даден ВЕП потребителя да добавя допълнителни графични и други елементи и езикови конструкции? Каква е сложността на процеса?
- Преизползваемост – по какъв начин е възможно части от даден ВЕП да бъдат използвани отново в друг. Може ли общи части от ВЕП да бъдат отделени в самостоятелни компоненти? Има ли разделение на отговорностите? Каква е сложността на процеса?
- Поддръжка – по какъв начин ВЕП ще се поддържа технически. Колко реда изходен код е необходим за изграждане на един ВЕП? Има ли добра документация и коментари в изходния код? Има ли разпределение на отговорностите? Каква е сложността на процеса?
- Достъпност – по какъв начин проектът може да се използва. Изходният му код свободно ли се разпространява? На какви платформи може да се изпълнява? Каква е сложността за добавяне на потребителски код в хранилището?

Изхождайки от добре познатата теория за контрол и оптимален контрол [65, стр. 15-20], процесът на разработка на приложенията може да се представи като опростена система със затворен цикъл. Затворената система трансформира изискванията

за програмата (в частност ВЕП) на разбираем за разработчика, Р, език ($T_1$). Разработчикът алгоритмизира изискванията ($T_2$) на език за програмиране (ЕП), а след това верига инструменти, ВИ, ги превеждат ($T_3$) във вид, подходящ за изпълнение, И.




Фигура 2.2: Интерактивност и динамичност.

В терминологията на теорията за контрол Р (и евентуално екипът на Р) е контролерът, а контролираният процес е съвкупността от $T_2$, ВИ и $T_3$. Обратната връзка представлява екип, който извършва наблюдения и измервания (често Р), и ако се забележат отклонения от "оптималната" траектория за реализиране на проекта, се предприемат съответните мерки. Времето за осъществяване на обратна връзка се нарича време за реакция. Фигура 2.2 онагледява разсъжденията.

Времето за реакция основно се доминира от ВИ, защото $T_2$ и $T_3$ са на практика инварианти, а в случаите, когато не са техните времена, са пренебрежимо малки. Веригата инструменти осигурява превода от ЕП до езика на КС. В зависимост от семантичните особености на езика, части от превода могат да се извършат преди изпълнението, правейки го по-ефективно.

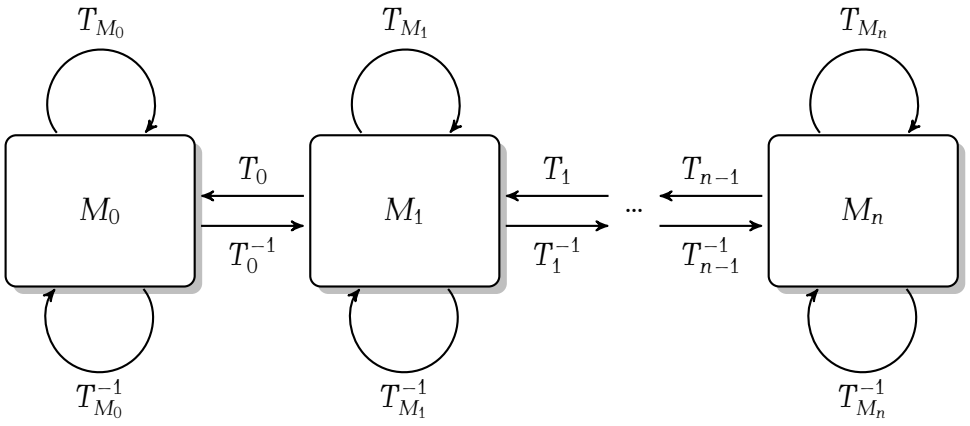
В този ред на мисли, термините *динамичност* и *интерактивност* могат да се въведат като различни реализации на механизма за реакция при незадоволителни крайни резултати от контролирания процес. Интерактивни са ЕП, чиито ВИ позволяват редакция по време на изпълнение. Динамични са ЕП, които намаляват времето за реакция. Обикновено реализацията на динамичен ЕП "отлага" някои стъпки, които се извършват от ВИ предварително и ги извършват по време на изпълнение. Затова се наблюдава загуба на ефективност при динамичните ЕП.

### 2.2.2 Програмата като (когнитивен) модел

Програмният език е нотация, чрез която хората описват изчисленията, които трябва да бъдат изпълнени от машина. Програмният език и нотацията са резултат от много по-сложен мисловен процес. Програмата е предписание или алгоритъм, описващ част

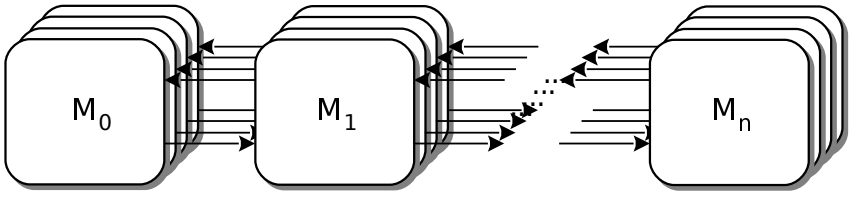
от реалния свят и представлява модел. От тази гледна точка, програма е едновременно когнитивен модел (за разработчика) и модел на изпълнение (за КС) за намиране на решение на даден проблем. Моделират се последователността от стъпки, които трябва да се извършат за достигане на задоволително решение.

Създаването на многомоделна архитектура, показана на фигура 2.3, обобщава трансформирането на софтуерния модел от високо ниво до модел, изпълним от виртуално изчислителна машина (ВИМ). В зависимост от целите, етапите (междинните модели), през които преминава трансформацията, може да са произволен брой.




Фигура 2.3: Взаимодействия между представянията

Гъвкавостта на мултимоделната архитектура е емпирично показана в [72], [73] и [74]. Тя позволява конструиране на разнородни системи, имащи множество качествено различни представяния и семантика. Цитираната библиография описва оптимизационна система, която използва множество представяния на програмата, за да постигне максимално добри оптимизационни резултати. Други емпирични изследвания в [75] и [76] конструират допълнителни модели към оптимизационната система, които са визуални. Сложността на представянията на оптимизационната система се намалява, като се предоставят визуални модели, подчертаващи някои трудно забележими взаимовръзки.

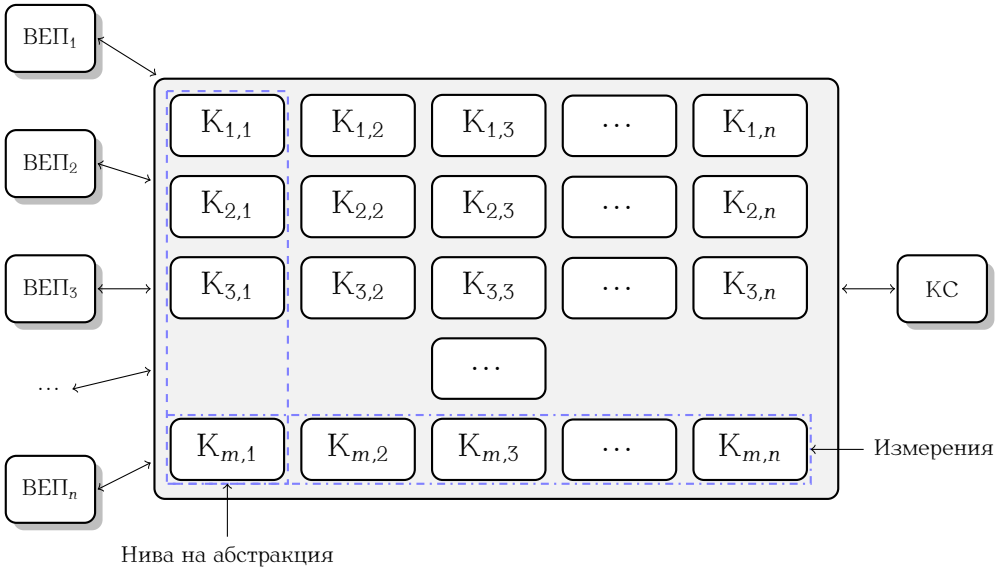



Фигура 2.4: Генерализация на взаимодействията между представянията.

Ако разгледаме дясната страна на фигура 2.3 се вижда, че $M_n$ може да бъде както сорс код от някакъв текстов ЕП, така и произволно визуално представяне, например UML

клас диаграма. За да включи многоканалната информация представена от визуалните модели се налага генерализация на 2.3 във фигура 2.4. Всеки канал на информация е моделиран като отделна верига от модели и евентуални техни трансформации.

Матричното представяне на схема 2.4 илюстрира някои структурни характеристики на архитектурата.




Фигура 2.5: Концептуална архитектура на системата за създаване на отворени, интерактивни и динамични ВЕП.

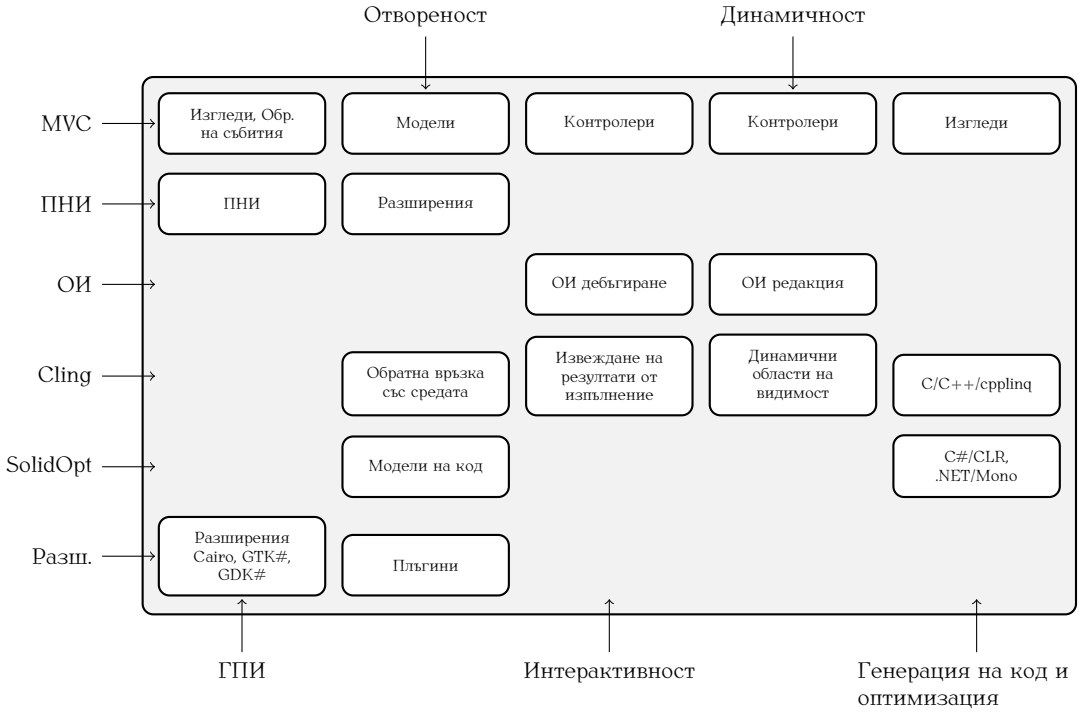
Структурно системата е разделена на нива на абстракция и проблемни измерения. Всеки елемент на матрицата (фигура 2.5) представлява логически отделена реализационна единица. Всяка реализационна единица в съответното измерение обособява точни отговорности и дефинира комуникационни интерфейси, чрез които взаимодейства с останалите. Тя може да бъде клас, съвкупност от класове, компонент или библиотека. Допустимо е, реализационните единици да могат да се разпростират в повече от едно ниво на абстракция и на повече от едно измерение, но само при необходимост. По-нататъшното разделяне на отговорностите служи за допълнително ограничаване на сложността на системата при разработка и работа с нея.

## 2.3 Глава 4 – Архитектура и прототипна реализация на метаинструментариума

*Обсъжда архитектурата и реализацията на метаинструментариума за създаване на ОИДВЕП, SolidV. Обосновава архитектурните решения, като детайлизира техническите особености на реализацията. Обсъждат се реализационните*

*особености на обобщения архитектурен шаблон Модел-Изглед-Контролер. Разглеждат се и важните съставни компоненти на SolidV като междумоделния оптимизатор SolidOpt и динамичния интерактивен интерпретатор Cling.*

Един широко приложим инструментариум за създаване на визуални среди би трябвало да поддържа лесно добавяне на множество функционалност от разнообразни приложни области. Архитектурата му трябва да бъде гъвкава, за да може да вмести функции от разнородни проблемни области, важни за създаването на произволен ВЕП. Специфичната функционалност, която прави дадена среда за предпочитане пред друга, прави задачата на инструментариума да изглежда нетривиална.




Фигура 2.6: Реализирани компоненти в SolidV.

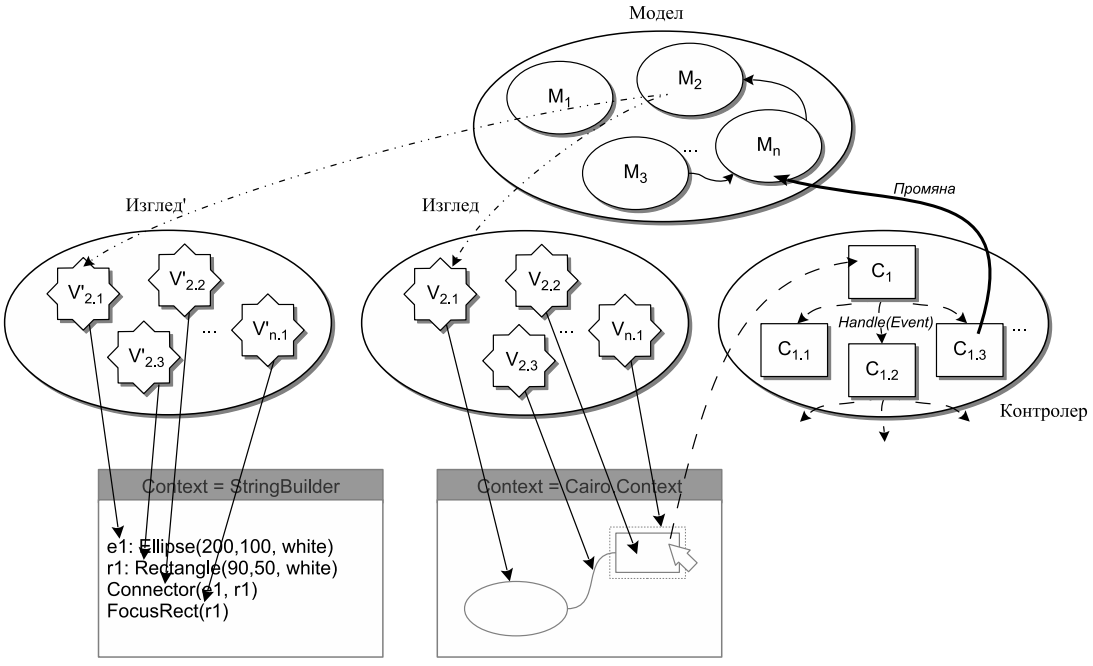
Разнородните задачи предполагат освен нивата на абстракция и специфика на проблемните области, накратко измерения (фигура 2.5). Например в измерението "превод" има няколко нива на абстракция като съпоставяне на модели, генериране на код и абстрактно синтактично дърво. Аналогично в измерението ГПИ (на фигура 2.6) – потребителски-настройваеми изгледи (ПНИ), обработка на събития и визуални представяния.

Фигура 2.6 онагледява реализираните от автора елементи във всяко едно от важните проблемни направления, описани по хоризонтала. По вертикала, низходящо са онагледени нивата на абстракция на съответните реализационни елементи. В следващите

няколко секции ще разгледаме няколко реализирани компонента от високо и ниско ниво в различни проблемни измерения.

### 2.3.1 Обобщен шаблон Модел-Изглед-Контролер

Архитектурният шаблон Модел-Изглед-Контролер (Model-View-Controller, MVC) [49] се състои от три вида класове. Моделът описва данните, изгледът е (не е задължително върху екран) представянето на данните, а контролерът дефинира начина, по който системата реагира на (потребителски) заявки. MVC разделя тези три отговорности и предоставя добре дефиниран комуникационен протокол между тях, чрез въвеждане на ниво на индирекция (фигура 2.7). Разделението на отговорностите предоставя на разработчиците на библиотеки и инструментариуми начин, по който да се справят със сложността на проектите. Метаинструментариумът използва MVC, защото той се е наложил като начин на мислене при съвременните разработчици. SolidV използва, разширява и обобщава концепциите, описани в [80]. Предимството на тази архитектура е, че налага комуникацията между измеренията на ВЕП да се осъществява чрез добре структурирани и дефинирани слоеве и комуникационни канали.




Фигура 2.7: MVC шаблонът в SolidV

Фигура 2.7 илюстрира как да се реализира по-гъвкава логика за модел, изглед и контролер. Предлагаме множество модели, които представят данните в разширими, йерархични (ако е необходимо) слоеве. Основният слой (в модела) е съставен от данни, които са изключително важни за работата на системата. Над него има слой, който

съдържа данни, които са важни за системата: например информация за селекция на елементи. Следващ слой е визуализационният, който може да бъде статичен или динамичен. Динамичният визуализационен слой позволява на потребителите да взаимодействат с него и да внасят промени. Контролерите взаимодействат с моделите и така реализират цялостното поведение на системата. Обикновено изгледът е отговорен за визуализацията на текущото състояние на даден модел. В нашата архитектура изгледът има по-обширни цели и отговорности. Той може да генерира код (Изглед' на фигура 2.7) на базата на модела и така да дефинира семантиката на ВЕП. Също така изгледите могат да наложат допълнителна информация върху "стандартната" визуализация като информация за дебъгиране и профайлинг.

Генерализираната схема 2.5 обобщава нивата на абстракция във вериги на абстракция за всяко измерение, т.е. всяко измерение има своя собствена верига и съответно логическа изолация. От друга страна MVC позволява разделяне на отговорностите за всяко измерение, т.е. всяка част от веригата е логически изолирана в зависимост от съответните отговорности.

### 2.3.2 Обектен инспектор

Повечето ВС трябва да предоставят визуални компоненти за разглеждане и редактиране на свойствата на различни визуални обекти. В зависимост от контекста и типа на данните, тяхната визуализация може да бъде направена от специализирани редактори. Така се постига консистентност при предаването на визуална информация.

Метаинструментариумът SolidV реализира обектен инспектор (ОИ) и лесни за вграждане в различни елементи редактори. Например математическите уравнения могат да бъдат разглеждани в обектния ОИ в математическа нотация. Реализацията на обектният инспектор в SolidV позволява редактирането на свойства по време на изпълнение, позволяващо по-добра динамика и интерактивност. Промените се правят директно върху представянето на стойностите в паметта, използвайки системата за интроспекция, поддържана от езика C#. Това също позволява на ОИ да бъде използван по време на дебъгиране.

### 2.3.3 Мултимоделна оптимизационна система SolidOpt

Използването на множество модели с ясни и обозрими механизми за трансформация би позволило много по-фино транслиране и оптимизиране на компютърните

програми. Тази идея е основополагаща за разработения мултимоделен оптимизационен инструментариум SolidOpt. Описаните концепции бяха приложени за реализиране на мултимоделна оптимизационна система, *SolidOpt*, в проблемната област. Теоретичният модел предполага добра модулност, която позволи много от работата да бъде извършена паралелно.

Оптимизационната система е проектирана да работи отдолу нагоре, т.е. може да оптимизира приложения, реализирани на CLR(ВИМ за .NET/Mono) без да е необходим техният изходен код. За целта SolidOpt може да конструира редица представяния, удобни за прилагане на различни оптимизационни методи и стратегии. Те биват категоризирани в няколко класа: модели на потоци (данни или инструкции), модели на кода и оптимизационни методи.

### 2.3.4 Интерактивен интерпретатор Cling

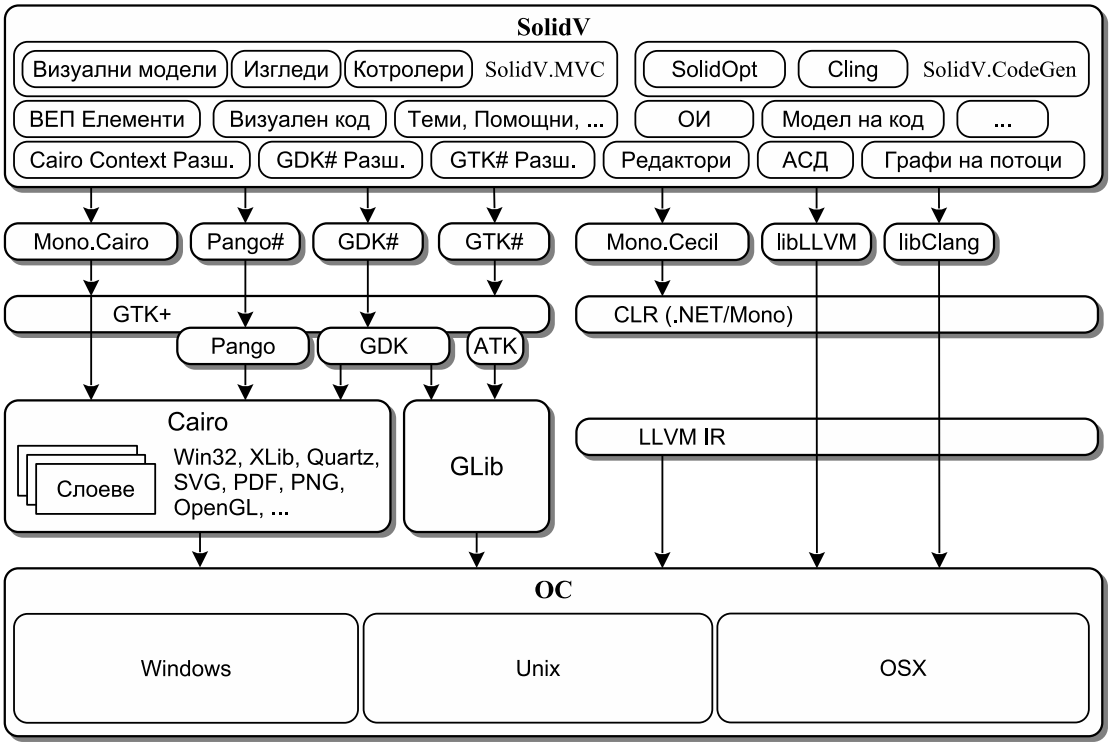
SolidOpt дава добра база за оптимизация и генерация на код за изпълнение, съответстващ на семантиката на визуалните модели. Инструментариумът може да се използва за генерация на код, работещ на CLR. Едно от предимствата е, че дава добра интеграция на ВЕП със съществуващи библиотеки за тази платформа.

Аналогично трябва да се предостави механизъм за генерация за C/C++. Има голямо количество софтуер, реализиран само на C/C++, но използването му от различни ЕП е сложно поради спецификите на езиците и разминаването между основните им концепции. Разработката на C/C++ интерпретатор ще позволи използване на софтуер, написан на съответният език, но и ще предостави достъп до изключително съвременни технологии и оптимизации.

Използването на Cling [101] като компонент в прототипната реализация има няколко предимства. Първо, той скъсява веригата инструменти, илюстрирана на фигура 2.2. Скъсяването на веригата инструменти повишава динамичността и интерактивността при комуникацията със средата. Второ, генерацията на код от метаинструментариума може да става в контекста на C/C++. При търсене на по-добра ефективност съответният ВЕП би могъл да избере генерация на C++ или някое друго по-ниско представяне, достъпно от интерпретатора. Cling може да се разглежда като алтернатива SolidOpt в контекста на C/C++ и оптимизаторът LLVM [98]. Трето, интерпретаторът позволява преизползването на много голямо количество библиотеки, съвместими със C/C++. Четвърто, върху Cling, SolidV може да конструира инфраструктура за дебъгиране и профайлинг на ВЕП.

### 2.3.5 Използвани технологии

Избраните технологии и тяхната връзка с архитектурата на прототипната реализация може да се изобрази схематично чрез Фигура 2.8.




Фигура 2.8: Използвани технологии от архитектурата и реализацията на SolidV

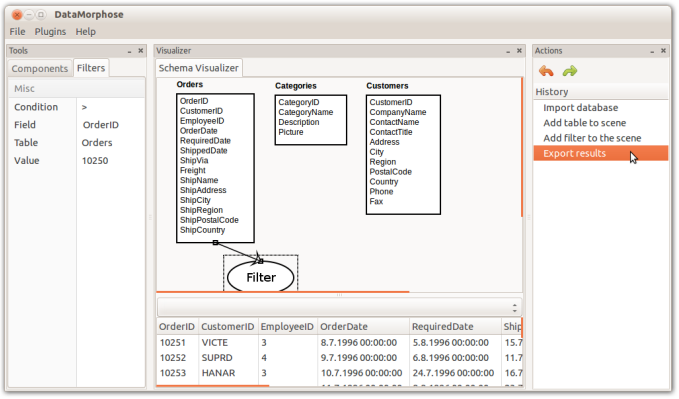
## 2.4 Глава 5 – Приложения и резултати

*Главата описва проекти, които използват прототипната реализация на метаинструментариума. Разглежда проблемната област на приложенията и разкрива детайли за тяхната реализация. Прилага изображения от работещите визуални системи. Демонстрира SolidIDE и SolidReflector – системите, използващи най-много от възможностите на прототипа. Описва самостоятелното използване на някои компоненти от SolidV. Представя приложението на интерактивния C/C++ интерпретатор Cling в областта на физиката на високите енергии.*

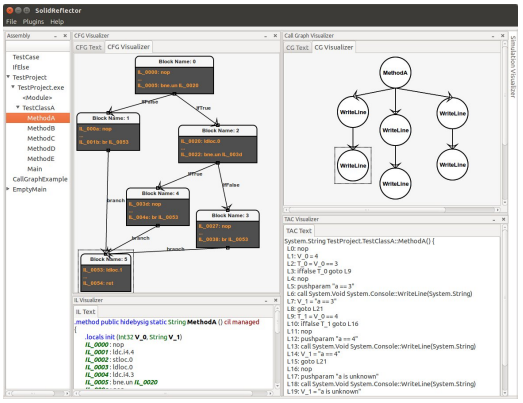
### 2.4.1 ОИДВЕП SolidIDE, SolidReflector, DataMorphose

SolidIDE, разработен от автора, представлява прототип на плъгин-базирана визуална среда за разработка и използване на ВЕП. Тя дава възможност за използване на

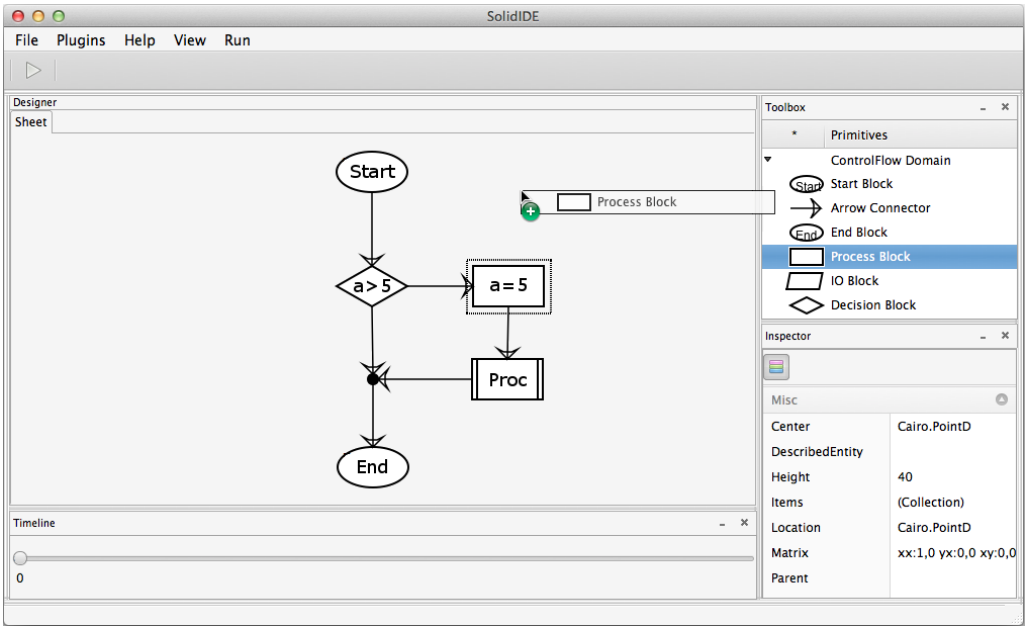
множество ВЕП едновременно. Всеки ВЕП може да бъде реализиран като отделно разширение на средата и да бъде зареден динамично. Това позволява на потребителите да изследват различните визуални елементи и да реализират комуникация между заредените ВЕП.




(а) DataMorphose – Визуален анализ и трансформации на данни




(б) SolidReflector – Визуален, интерактивен, статичен и динамичен анализ на .NET приложения




(в) SolidIDE – интегрирана среда за разработка

Фигура 2.9: ОИДВЕП SolidIDE, SolidReflector, DataMorphose

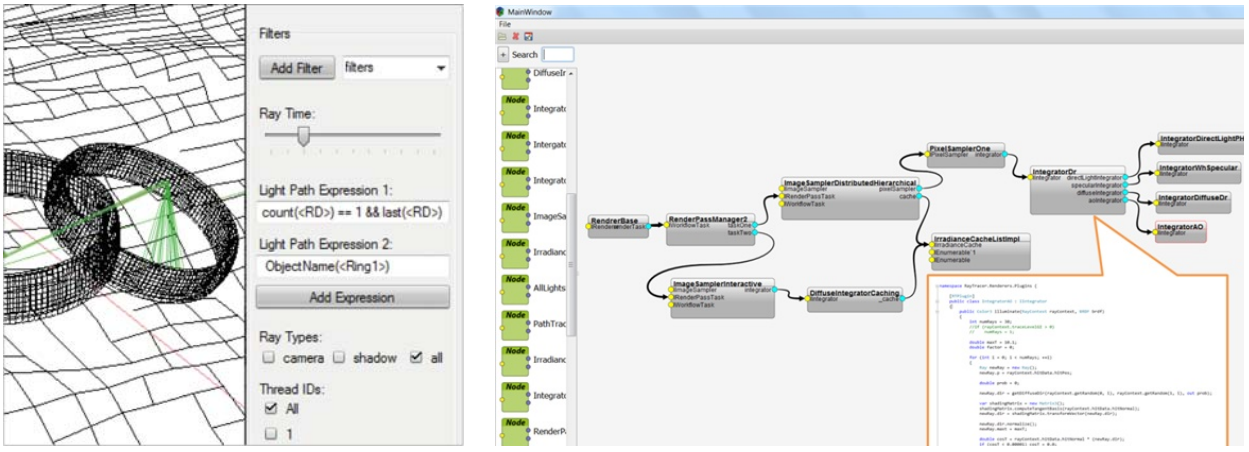
SolidReflector [75, 76], чиято разработка е ръководена от автора, на фигура 2.9(б) е инструмент за мултимоделен интерактивен анализ и декомпилация, който е способен да декомпилира .NET/Mono изпълними файлове.

DataMorphose [107, 108], чиято разработка е ръководена от автора, на фигура 2.9(а) реализира ВЕП, ориентиран към потока от данни, подпомагащ потребители без много експертни познания в областта на миграцията на данни.

### 2.4.2 Визуално конструиране на системи за глобално осветяване

Системата за модулно, разширяемо глобално осветяване, предмет на докторската дисертация [109], [110] използва части от метаинструментариума SolidV. При конфигурирането на модули, реализиращи различни алгоритми за глобално осветяване, трябва да се вземат предвид много ограничения като тяхната съвместимост. ВЕП, улеснява процеса по конфигурирането на модули и изграждане на готово за използване приложение (фигура 2.10(б)). Авторите използват метаинструментариума SolidV, като реализират ВЕП, базиран на ограничения.

Инструментът предоставя дружелюбна потребителска среда, която е лесно разбираема за неексперти. Поддържа механизъм за влачене-и-пускане, като всеки блок представлява алгоритъм и се характеризира с входни и изходни пинове, моделиращи входните и изходните данни на алгоритъма. Връзките между блоковете се осъществяват на базата на анализ дали входните и изходните данни между двата блока са съвместими. Конструираната визуална програма се превежда до езика на конкретния инструментариум, чрез генерация на код на ЕПВН IronPython.




(а) Домейн-специфичен визуален дебъгер. (б) Визуално конструиране на система за глобално осветяване.

Фигура 2.10: Визуално конструиране и дебъгиране на система за глобално осветяване

Крайният резултат на системата за модулно, разширяемо, глобално осветяване представлява фотореалистично изображение [111]. При неправилна работа на някои от алгоритмите се искат редица усилия да се намери конкретният проблем, използвайки стандартни инструменти за дебъг. Една от причините е, че крайният резултат се различава качествено от междинните му представяния, използвани от алгоритмите. Още повече, че е много трудно да се идентифицира съответният алгоритъм, който не работи както се очаква.

Системата предоставя и домейн-специфичен визуален дебъгер, базиран на SolidV. Дебъгерът оставя информация под формата на следи (traces) в специален "дебъг" модел, който може да бъде насложен върху генерираното изображение и ако качеството се влоши, може лесно да покаже множеството от модули, участващи в генерацията на некоректната част (фигура 2.10(a)). Промените (и техните делти) могат да се следят в реално време. Използвайки обектния инспектор, могат да се прилагат различни механизми на филтрация, така че дефектите да могат да се анализират по-лесно.

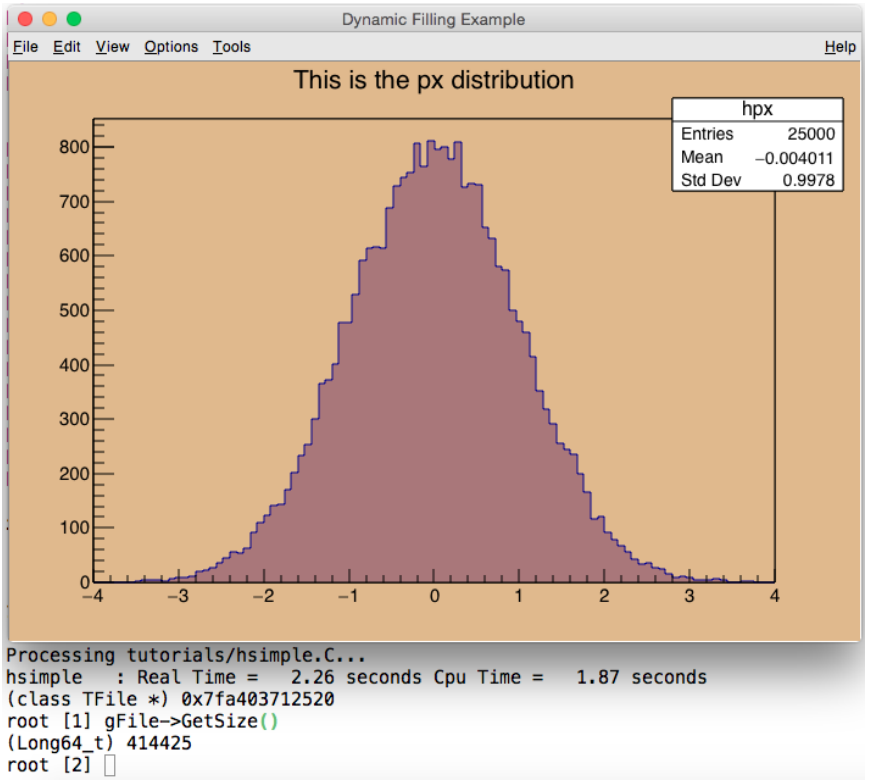
### 2.4.3 Интерпретаторът Cling в областта на физиката

Бързата разработка на приложения [112] става все по-важна в областта на науката и индустрията. Възможността за бърза разработка на софтуерни прототипи и приложения, които онагледяват концепции се превръща в предусловие за успеха на много проекти. Създаването на прототипи е ефективен начин за разбиране на изискванията, намаляване на сложността на проблемите и предоставяне на възможност за ранна валидация на дизайна и реализацията на системата. Обаче, всичко това зависи от времето, консумирано от цикъла редакция-компилация-изпълнение по време на разработка. Езиците, които се компилират още повече затрудняват този процес, тъй като времето отделено за компилация и свързване (linking) е значително повече.

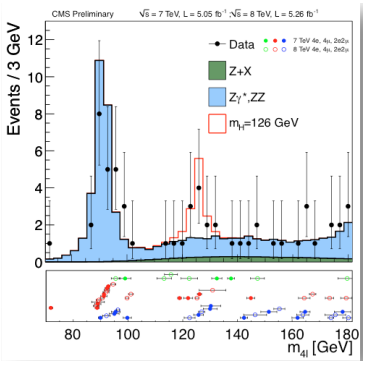
Основно реализиран на C++, ROOT [113] е огромен инструментариум, съставен от библиотеки за съхраняване, анализиране, обработване и визуализиране на данни. По скорошни оценки ROOT се използва за съхранение на повече от 180 PB данни, което го незаменима част в областта на физиката на високите енергии (ФВЕ). ROOT е проектиран да работи с огромни обеми данни и се използва широко дори извън областта на ФВЕ като високо-честотно търгуване, биология и астрономия.

Повечето от неговите потребители не са експерти и затова инструментариумът трябва да им предложи начин за разработка на техните приложения бързо и лесно. За да постигне това, ROOT използва техниките на бързата разработка на приложения. Компонентът, отговорен за реализирането на подобен стил на изследователско програмиране е неговият интерактивен C++ интерпретатор.

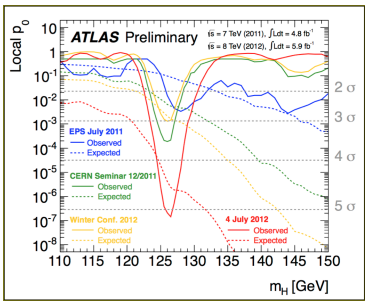
Интерактивният C++ интерпретатор Cling, разработен по време на настоящата работа, успя да замени до тогава използвания интерпретатор. Това е външна атестация за добрата преизползваемост на компонентите на метаинструментариума SolidV. Новите версии на ROOT с Cling (ROOT6, фигура 2.11) ще бъдат основна част от програмата на CERN и експериментите върху LHC [114] през 2015-2020 г.




(а) ROOT и неговия интерпретатор Cling.




(б) Диаграма от експеримента CMS.




(в) Диаграма от експеримента ATLAS.

Фигура 2.11: ROOT и диаграми на генерираните от него визуализации на следи от бозона на Хигс.

## 2.5 Глава 6 – Заключение

Дисертационният труд разглежда обстойно и предлага аргументирани решения на всяка една от поставените в секция 1.2 – Цели и задачи на дисертационния труд. По време на изследванията на проблемната област са направени няколко приноса, описани в следващите подсекции.

По време на работата в проблемната област са разкрити и други важни подобласти на тематиката и са предложени аргументирани решения. Тези допълнителни разработки улесняват работата по основните цели на дисертационния труд и значително разширяват областта на приложимост.

Също така, трябва да се отбележи и приносът в обучението и кариерното развитие на студенти. Докторската работа подпомогна обособяването на подпроекти, подходящи за студенти и дипломанти. Бяха направени редица подобрения по съществуващи учебни курсове. Бяха разработени и нови, съвременни курсове, предназначени за студенти в областта на информатиката и информационните технологии.

### 2.5.1 Резултати и приноси

При решаването на поставените цели бяха получени следните основни научни, научно-приложни и приложни приноси към тематиката:

| # | Резултати | Цел | Секция | Публикации |
|---|---|---|---|---|
| 1 | Предложен е модел (концепция) за създаване на ОИДВЕП | **Ц1** | Глава 3 | **С1** |
| 2 | Предложена е архитектура на софтуерения метаинструментариум за създаване на ОИДВЕП | **Ц1** | Глава 3 | **С1** |
| 3 | Разработен е прототип на метаинструментариум, *SolidV*, реализиращ основните идеи на дисертационния труд | **Ц2** | Глава 4 | **С5** |
| 4 | Реализиран е прототип на ВСП, *SolidIDE*, базирана на модела (и използваща предложената архитектура) | **Ц3** | Глава 5 | - |
| 5 | Показани са примери за преизползване на компонентите за разработка на други ВЕП (SolidReflector, DataMorphose и др.). Показани са примери за преизползване на компонентите в други области | **Ц3** | Глава 5 | **С2**, **С3**, **С4** |

Таблица 2.2: Връзка между цели и приноси.

### 2.5.2 Публикации по дисертационния труд

**С1** **Vassilev V.**, Penev A., Vassilev M., "SolidOpt – A Multi-Model Software Optimization Framework", International Journal of Computer Science Issues, Issue March 2nd 2015, pp. 32-41, [ISSN (Print): 1694-0814, ISSN (Online): 1694-0784];

**С2** **Vassilev V.**, Vassilev M., Petrova P., "SolidReflector: A Multistage, Interactive Decompilation Framework", in Proceedings of the International Conference "From DeLC to Velspace Plovdiv, 26-28 March 2014, pp. 49-58 [ISBN 0-9545660-2-5];

**С3** **Vassilev V.**, Petrova P., Vassilev M., "DataMorphose: An Interactive Data Migration Framework", in Proceedings of the International Conference "From DeLC to Velspace Plovdiv, 26-28 March 2014, pp. 59-68 [ISBN 0-9545660-2-5];

**С4** **Vassilev V.**, Naumann A., Canal Ph., Russo P., "Cling – The New Interactive Interpreter for ROOT 6", in Proceedings of Computing in High Energy Physics, New York, NY, USA, 2012, article number: 052071, 10 pp., [V Vasilev et al 2012 J. Phys.: Conf. Ser. 396 052071 doi:10.1088/1742-6596/396/5/052071];

**С5** Penev A., Dimov D., **Vassilev V.**, "Applying Model-View-Controller in Geometric Modelling Methodology", in Proceedings of the International conference in Computer Science, Varna, Bulgaria, July 2008, pp. 285-294 [ISSN 1313-4345].

### 2.5.3 Доклади на конференции и семинари

**П1** Clad – Automatic Differentiation with Clang, prof. Kahan research group, Lawrence-Berkeley National Laboratory, Berkeley, CA, USA, November 2014, представящ;

**П2** Native Language Integrated Queries with CppLINQ in C++, ACAT 2014, Prague, Czech Republic, September 2014, представящ;

**П3** Clad – Automatic Differentiation Using Cling/Clang/LLVM, ACAT 2014, Prague, Czech Republic, September 2014, представящ;

**П4** Interactive, Introspected C++ at CERN: CppNow 2013, Aspen Center for Physics, Aspen, CO, USA, May 2013, представящ;

**П5** Cling – The New Interactive Interpreter for ROOT 6: CHEP 2012, New York University, New York, NY, USA, May 2012, представящ;

**П6** Introducing cling, a C++ Interpreter Based on clang/LLVM, Google TechTalk, Google, Zurich, Switzerland, 15 March 2012, участващ в представянето;

**П7** Implementing Dynamic Scopes in Cling: Invited Talk for the First LLVM Euro Developers' Meeting, London, England, September 2011, представящ;

**П8** The Role of the Multiple Notations in the Programming Languages. SolidOpt Notation Graph. Call Graphs, Control-Flow Graphs, Three-Address Representations and SSA, Invited Talk for the 12-2010 SolidOpt Developers' Meeting, Plovdiv, Bulgaria, December 2010, представящ;

**П9** Creating Cling, an Interactive Interpreter Interface for Clang: Axel Naumann, Philippe Canal, Paul Russo, Vasil Vasilev, Invited Talk for the 5-th LLVM Developers' Meeting, San Jose, CA, United States, November 2010, представящ;

**П10** A Framework for Automated Optimization of Software Applications (SolidOpt), Invited Talk for seminar Contemporary techniques for optimization of software applications, University of Plovdiv, Bulgaria, November 2009, представящ.

### 2.5.4 Обучение и участия в проекти

Важна част от докторските програми е работата със студенти и развитието и подобряването на университетските програми за обучение. Чл. 25. (1) от ПРАС на ПУ насърчава приноса на докторантите в това направление. В този смисъл настоящият

дисертационен труд допринесе за развитието на курсове и обучението на студенти на Факултет по математика и информатика при Пловдивски университет „Паисий Хилендарски“. Част от приносите са и към други международни университети и организации.

Разработени учебни курсове:

- Разработка на спец. курс *“Анализ и оптимизация на софтуерни приложения”*;
- Разработка и подобряване на материали по *“Компютърна Графика”* и *“Методи на транслация”*.

Участия в проекти в Пловдивски университет „Паисий Хилендарски“:

- Научен проект НИ15-ФМИИТ-004/24.04.2015 *“Иновативни фундаментални и приложни научни изследвания по компютърни науки, математика и педагогика на обучението”* с ръководител проф. д-р Снежана Гочева-Илиева, Фонд “Научни изследвания”, 2015 г.
- Научен проект НИС14-ФМИИТ-002/26.03.2014 *“Разработка на паралелни алгоритми с приложения в научните изследвания и обучението по математика и информатика”* с ръководител доц. д-р Христо Крушков, Фонд “Научна инфраструктура”, 2014 г.

Участия в проекти в европейски център за ядрени изследвания (CERN):

- ROOT – ROOT е инструментариум за обработка на данните, разработен в CERN, централна част на изследователската дейност на ФВЕ. Всеки ден хиляди физици използват приложения базирани на ROOT, за да анализират експериментални данни или да извършват симулации. Използва се от почти всички експерименти на LHC и съхранява техните данни (около 180 PB). Авторът е основен разработчик на голяма част от подсистемите в ядрото на ROOT.

## 2.6 Изводи

Основните идеи в настоящия дисертационен труд бяха плод на усилена работа преди и по време на обучението за придобиване на образователна степен “доктор”. Идеите за прилагането на многомоделна архитектура се подобряваха с времето като това може да се забележи в дипломните работи, съответно за придобиване на образователна степен бакалавър [72] и магистър [73] на автора.

На базата на тези основи дисертационният труд успя да предложи генерално и добре реализирано решение на поставените цели. Успя да реализира и усъвършенства различни компоненти, които намериха приложение дори извън контекста на работата и академичната общност. Част от решенията са публикувани на международни конференции, а друга е в процес на рецензия за публикация. Предложената реализация на метаинструментариума е използвана за създаването на различни ВС от автора и други потребители. Реализираните приложения са качествено различни:

- SolidIDE – ВС, онагледяваща функционалността на SolidV;
- SolidReflector – ВЕП, ориентиран към потока инструкции;
- DataMorphose – ВЕП, ориентиран към потока данни;
- Визуален конфигуратор – ВЕП, базиран на ограничения за интерактивно създаване на приложения за трасиране на лъчи;
- Домейн-специфичен визуален дебъгер – дебъгер, улесняващ откриването на проблеми в домейн-специфични условия и показващ гъвкавостта и широката приложимост на метаинструментариума.

**На базата на представените резултати може да се заключи, че целите и задачите на дисертационния труд са изпълнени успешно.**

# Библиография

[1] Sutherland, I. E. Sketchpad: A Man-machine Graphical Communication System. В. *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference*. Detroit, Michigan: ACM, 1963, с. 329–346. AFIPS '63 (Spring).

[2] Victor, B. *Stop Drawing Dead Fish* [онлайн]. 2012 [прегл. 2015-06-01]. ⟨`http://san-francisco.siggraph.org/2012/05/25/stop-drawing-dead-fish/`⟩.

[3] Aho, A. V. и др. *Compilers: Principles, Techniques, and Tools (2Nd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006. ISBN 0321486811.

[4] Dahl, O. J. и др. (ред.). *Structured Programming*. London, UK, UK: Academic Press Ltd., 1972. ISBN 0-12-200550-3.

[5] Smith, D. *Pygmalion: A Computer Program to Model and Stimulate Creative Thought*. 1977. Interdisciplinary systems research. ⟨`http://books.google.bg/books?id=32MoAQAAIAAJ`⟩. ISBN 9783764309282.

[6] Myers, B. A. Visual Programming, Programming by Example, and Program Visualization: A Taxonomy. В. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Boston, Massachusetts, USA: ACM, 1986, с. 59–66. CHI '86. ISBN 0-89791-180-6.

[7] Daintith, J. и др. *A Dictionary of Computing*. 6. изд. New York, NY, USA: Oxford University Press, Inc., 2008. ISBN 0199234019, 9780199234011.

[8] Rozenberg, G. (ред.). *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1997. ISBN 98-102288-48.

[9] Burnett, M. M. и др. *A Classification System for Visual Programming Languages*. 1994. ⟨`http://www.sciencedirect.com/science/article/pii/S1045926X84710159`⟩.

[10] Myers, B. A. Taxonomies of Visual Programming and Program Visualization. *J. Vis. Lang. Comput.* 1990, т. 1, № 1, с. 97–123. ISSN 1045-926X.

[11] Green, T. R. G. и др. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing*. 1996, т. 7, № 2, с. 131–174.

[12] Christensen, C. On the Implementation of AMBIT, a Language for Symbol Manipulation. *Commun. ACM*. 1966, т. 9, № 8, с. 570–573. ISSN 0001-0782.

[13] Ellis, T. и др. *The GRAIL Project: An Experiment in Man-machine Communication*. 1969. Memorandum (Rand Corporation). ⟨http://books.google.bg/books?id=P93KPAAACAAJ⟩.

[14] Pietrzykowski, T. и др. The Programming Language PROGRAPH: Yet Another Application of Graphics. В. *The Programming Language PROGRAPH: Yet Another Application of Graphics*. 1983, с. 143–145.

[15] Meyer, R. M. и др. Towards a Better Visual Programming Language: Critiquing Prograph's Control Structures. *J. Comput. Sci. Coll.* 2000, т. 15, № 5, с. 181–193. ISSN 1937-4771.

[16] Van Reeth, F. и др. IGIP: a framework towards open-ended visual programming. В. *Visual Languages, 1988., IEEE Workshop on*. 1988, с. 239–247. ⟨http://dx.doi.org/10.1109/WVL.1988.18034⟩. ISBN 0-8186-0876-5.

[17] Visual Solutions Inc. *VisSim A graphical language for simulation and model-based embedded development* [онлайн]. [прегл. 2015-05-03]. ⟨http://www.vissim.com/downloads/data_sheets.html⟩.

[18] Kurlander, D. и др. Inferring Constraints from Multiple Snapshots. *ACM Trans. Graph.* 1993, т. 12, № 4, с. 277–304. ISSN 0730-0301.

[19] Bentrad, S. и др. Visual Programming and Program Visualization–Towards an Ideal Visual Software Engineering System. *IJIT-ACEEE International Journal on Information Technology*. 2011, т. 1, № 3, с. 56–62.

[20] Mitkin, S. *DRAKON-Erlang: Visual functional programming* [онлайн]. 2012 [прегл. 2015-02-19]. ⟨http://drakon-editor.sourceforge.net/drakon-erlang/intro.html⟩.

[21] Taentzer, G. и др. Amalgamated Graph Transformations and Their Use for Specifying AGG - an Algebraic Graph Grammar System. В. *Proceedings of the International Workshop on Graph Transformations in Computer Science*. London, UK, UK: Springer-Verlag, 1994, с. 380–394. ISBN 3-540-57787-4.

[22] Taentzer, G. AGG: A graph transformation environment for modeling and validation of software. В. *Applications of Graph Transformations with Industrial Relevance*. 2004, с. 446–453.

[23] Rekers, J. и др. Defining and parsing visual languages with layered graph grammars. *Journal of Visual Languages & Computing*. 1997, т. 8, № 1, с. 27–55.

[24] Schürr, A. и др. Handbook of Graph Grammars and Computing by Graph Transformation. В Ehrig, H. и др. (ред.). *Handbook of Graph Grammars and Computing by Graph Transformation*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1999, с. 487–550. ISBN 981-02-4020-1.

[25] Soi, S. и др. Conceptual Development of Custom, Domain-Specific Mashup Platforms. *ACM Trans. Web*. 2014, т. 8, № 3, с. 14:1–14:35. ISSN 1559-1131.

[26] YQL. *Yahoo!Pipes* [онлайн]. [прегл. 2015-06-12]. ⟨http://pipes.yahoo.com/pipes/⟩.

[27] Software AG. *Presto Wires* [онлайн]. [прегл. 2015-05-11]. ⟨http://www.jackbe.com/products/wires.php⟩.

[28] Baycastle Software Ltd. *DataSlave* [онлайн]. [прегл. 2015-05-14]. ⟨http://www.baycastle.co.uk/V2/DataSlave/DataSlave.htm⟩.

[29] Bresson, J. и др. OpenMusic: Visual Programming Environment for Music Composition, Analysis and Research. В. *Proceedings of the 19th ACM International Conference on Multimedia*. Scottsdale, Arizona, USA: ACM, 2011, с. 743–746. MM '11. ISBN 978-1-4503-0616-4.

[30] *KTechLab*. [онлайн]. [прегл. 2015-06-02]. ⟨http://sourceforge.net/projects/ktechlab/⟩.

[31] LLC, M. S. *OpenWire* [онлайн]. 2001 [прегл. 2015-05-08]. ⟨http://www.mitov.com/products/openwire⟩.

[32] McWhirter, J. D. и др. Escalante: An environment for the rapid construction of visual language applications. 1994.

[33] Minas, M. и др. DiaGen: A Generator for Diagram Editors Providing Direct Manipulation and Execution of Diagrams. В. *Proceedings of the 11th International IEEE Symposium on Visual Languages*. Washington, DC, USA: IEEE Computer Society, 1995, с. 203–. VL '95. ISBN 0-8186-7045-2.

[34] Köth, O. и др. Generating diagram editors providing free-hand editing as well as syntax-directed editing. В. *Proc. International Workshop on Graph Transformation*. 2000.

[35] Minas, M. Concepts and realization of a diagram editor generator based on hypergraph transformation. *Science of Computer Programming*. 2002, т. 44, № 2, с. 157–180. Special Issue on Applications of Graph Transformations (GRATRA 2000). ⟨http://www.sciencedirect.com/science/article/pii/S0167642302000370⟩. ISSN 0167-6423.

[36] Baresi, L. и др. Introducing Formal Specification Methods in Industrial Practice. В. *Proceedings of the 19th International Conference on Software Engineering*. Boston, Massachusetts, USA: ACM, 1997, с. 56–66. ICSE '97. ISBN 0-89791-914-9.

[37] Klein, P. и др. Constructing SDEs with the IPSEN Meta Environment. В. *Proceedings of the 8th International Conference on Software Engineering Environments (SEE '97)*. Washington, DC, USA: IEEE Computer Society, 1997, с. 2–. SEE '97. ISBN 0-8186-8019-9.

[38] Rumbaugh, J. и др. *Unified Modeling Language Reference Manual, The (2Nd Edition)*. 2004. ISBN 0321245628.

[39] Egyed, A. и др. Rose/Architect: A Tool to Visualize Architecture. В. *Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8 - Volume 8*. Washington, DC, USA: IEEE Computer Society, 1999, с. 8066–. HICSS '99. ISBN 0-7695-0001-3.

[40] Lédeczi, Á. и др. Composing Domain-Specific Design Environments. *Computer*. 2001, т. 34, № 11, с. 44–51. ISSN 0018-9162.

[41] Kelly, S. и др. MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. В. *Proceedings of the 8th International Conference on Advances Information System Engineering*. London, UK, UK: Springer-Verlag, 1996, с. 1–21. CAiSE ;96. ISBN 3-540-61292-0.

[42] *Eclipse Modeling Framework (EMF)*. [онлайн]. [прегл. 2015-06-01]. ⟨`http://www.eclipse.org/emf`⟩.

[43] *Graphical Modeling Framework (GEF)*. [онлайн]. [прегл. 2015-05-01]. ⟨`http://www.eclipse.org/gef`⟩.

[44] IBM Corporation. *Rational Rose XDE Modeler* [онлайн]. [прегл. 2015-04-05]. ⟨`http://www-306.ibm.com/software/awdtools/developer/modeler/`⟩.

[45] Jennings, G. J. Ř. *LabVIEW Graphical Programming*. 2nd. 2006. ISBN 0071451463.

[46] Goldberg, A. и др. Visual Object-oriented Programming. В Burnett, M. M. и др. (ред.). *Visual Object-oriented Programming*. Greenwich, CT, USA: Manning Publications Co., 1995, с. 3–20. ISBN 0-13-172397-9.

[47] Myers, B. A. и др. The Amulet Environment: New Models for Effective User Interface Software Development. *IEEE Trans. Softw. Eng.* 1997, т. 23, № 6, с. 347–365. ISSN 0098-5589.

[48] Flannery, L. P. и др. Designing ScratchJr: Support for Early Childhood Learning Through Computer Programming. В. *Proceedings of the 12th International Conference on Interaction Design and Children*. New York, New York, USA: ACM, 2013, с. 1–10. IDC '13. ISBN 978-1-4503-1918-8.

[49] Krasner, G. E. и др. A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80. *J. Object Oriented Program.* 1988, т. 1, № 3, с. 26–49. ISSN 0896-8438.

[50] Vlissides, J. M. и др. Unidraw: A Framework for Building Domain-specific. В. *Proceedings of the 2Nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology*. Williamsburg, Virginia, USA: ACM, 1989, с. 158–167. UIST '89. ISBN 0-89791-335-3.

[51] Schuckmann, C. и др. Designing Object-oriented Synchronous Groupware with COAST. В. *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*. Boston, Massachusetts, USA: ACM, 1996, с. 30–38. CSCW '96. ISBN 0-89791-765-0.

[52] Buchner, J. и др. HotDoc - A Flexible Framework for Spatial Composition. В. *Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*. Washington, DC, USA: IEEE Computer Society, 1997, с. 92–. VL '97. ISBN 0-8186-8144-6.

[53] Ferguson, R. и др. MetaMOOSE—an object-oriented framework for the construction of {CASE} tools. *Information and Software Technology*. 2000, т. 42, № 2, с. 115–128. ⟨`http://www.sciencedirect.com/science/article/pii/S0950584999000816`⟩. ISSN 0950-5849.

[54] Welch и др. *Practical Programming in Tcl & Tk*. 4. изд. 2003. ISBN 0130385603.

[55] Dewan, P. и др. Flexible User Interface Coupling in a Collaborative System. В. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New Orleans, Louisiana, USA: ACM, 1991, с. 41–48. CHI '91. ISBN 0-89791-383-3.

[56] Riazanov, A. и др. The Design and Implementation of VAMPIRE. *AI Commun.* 2002, т. 15, № 2,3, с. 91–110. ISSN 0921-7126.

[57] Zhang, D.-Q. и др. VisPro: A Visual Language Generation Toolset. В. *Proceedings of the IEEE Symposium on Visual Languages*. Washington, DC, USA: IEEE Computer Society, 1998, с. 195–. VL '98. ISBN 0-8186-8712-6.

[58] Grundy, J. и др. Visual specification of multi-view visual environments. В. *Visual Languages, 1998. Proceedings. 1998 IEEE Symposium on*. 1998, с. 236–243. ⟨`http://dx.doi.org/10.1109/VL.1998.706168`⟩.

[59] Phillips, C. и др. The Design of the Client User Interface for a Meta Object-Oriented CASE Tool. В. *Proceedings of the Technology of Object-Oriented Languages and Systems*. Washington, DC, USA: IEEE Computer Society, 1998, с. 156–. TOOLS '98. ISBN 0-7695-0053-6.

[60] Grundy, J. и др. Constructing component-based software engineering environments: issues and experiences. *Information and Software Technology*. 2000, т. 42, № 2, с. 103–114.

[61] El Kouhen, A. и др. Evaluation of Modeling Tools Adaptation. 2012. ⟨`https://hal.archives-ouvertes.fr/hal-00706701`⟩.

[62] Kern, H. и др. Towards a Comparative Analysis of Meta-metamodels. В. *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*. Portland, Oregon, USA: ACM, 2011, с. 7–12. SPLASH '11 Workshops. ISBN 978-1-4503-1183-0.

[63] Myers, B. A. и др. Survey on User Interface Programming. В. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Monterey, California, USA: ACM, 1992, с. 195–202. CHI '92. ISBN 0-89791-513-5.

[64] Daniel, F. и др. Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*. 2007, т. 11, № 3, с. 59–66. ISSN 1089-7801.

[65] Kirk, D. E. *Optimal control theory: an introduction*. 2012. ISBN 0486434842.

[66] Golin, E. J. *A Method for the Specification and Parsing of Visual Languages*. Providence, RI, USA: Brown University, 1992. UMI Order No. GAX92-04867.

[67] Golin, E. J. Parsing Visual Languages with Picture Layout Grammars. *J. Vis. Lang. Comput.* 1991, т. 2, № 4, с. 371–393. ISSN 1045-926X.

[68] Mace, S. Context-Driven Constraint Multiset Grammars with Incremental Parsing for On-line Structured Document Interpretation. В. *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2007, с. 442–446. ICDAR '07. ISBN 0-7695-2822-8.

[69] Marriott, K. Parsing Visual Languages with Constraint Multiset Grammars. В. *Proceedings of the 7th International Symposium on Programming Languages: Implementations, Logics and Programs*. London, UK, UK: Springer-Verlag, 1995, с. 24–25. PLILPS '95. ISBN 3-540-60359-X.

[70] Low, D. Protecting Java Code via Code Obfuscation. *Crossroads*. 1998, т. 4, № 3, с. 21–23. ISSN 1528-4972.

[71] Гочева С. *Лекции по Компютърни числени методи*. ⟨`http://www.fmi-plovdiv.org/evlm/DBbg/numanmenu/index.htm`⟩.

[72] Василев, В. *SolidOpt – Инструментариум за автоматизирани оптимизации на софтуерни приложения*. Дипломна работа. Пловдив, 2009. 99 л.

[73] Василев, В. *SolidOpt – Инструментариум за автоматизирани оптимизации на софтуерни приложения. Оптимизации, зависещи от приложната област*. Дипломна работа. Пловдив, 2010. 127 л.

[74] Vassilev, V. и др. SolidOpt–A Multi-Model Software Optimization Framework. *International Journal of Computer Science Issues*. 2015, т. 12, с. 32–41. ⟨`http://ijcsi.org/papers/IJCSI-12-2-32-41.pdf`⟩.

[75] Василев, М. *SolidReflector – Визуален, интерактивен, статичен и динамичен анализ на .NET приложения*. Дипломна работа. Пловдив, 2013. 87 л.

[76] Vassilev, V. и др. SolidReflector: A Multistage, Interactive Decompilation Framework. В. *SolidReflector: A Multistage, Interactive Decompilation Framework*. 2014, с. 49–58. ISBN 0-9545660-2-5.

[77] Varde, A. s. и др. Comparing Mathematical and Heuristic Approaches for Scientific Data Analysis. *Artif. Intell. Eng. Des. Anal. Manuf.* 2008, т. 22, № 1, с. 53–69. ISSN 0890-0604.

[78] Law, A. M. Statistical Analysis of Simulation Output Data: The Practical State of the Art. В. *Proceedings of the 36th Conference on Winter Simulation*. Washington, D.C.: Winter Simulation Conference, 2004, с. 67–72. WSC '04. ISBN 0-7803-8786-4.

[79] Glushkova, T. Adaptive Model for User Knowledge in the e-Learning System. В. *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*. Gabrovo, Bulgaria: ACM, 2008, с. 78:V.16–78:1. CompSysTech '08. ISBN 978-954-9641-52-3.

[80] Penev, A. и др. Applying Model-View-Controller in Geometric Modelling Methodology. В. *Applying Model-View-Controller in Geometric Modelling Methodology*. 2008, с. 59–66.

[81] Kara, D. Build vs. Buy-Maximizing the Potential of Components. *Component Strategies*. 1998, т. 1, с. 22–23.

[82] Batory, D. и др. The Design and Implementation of Hierarchical Software Systems with Reusable Components. *ACM Trans. Softw. Eng. Methodol.* 1992, т. 1, № 4, с. 355–398. ISSN 1049-331X.

[83] Senthil, R. и др. An improved component model for component based software engineering. *ACM SIGSOFT Software Engineering Notes*. 2007, т. 32, № 4, с. 9.

[84] Szyperski, C. *Component Software: Beyond Object-Oriented Programming*. 2nd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0201745720.

[85] Meyer, B. The Grand Challenge of Trusted Components. В. *Proceedings of the 25th International Conference on Software Engineering*. Portland, Oregon: IEEE Computer Society, 2003, с. 660–667. ICSE '03. ISBN 0-7695-1877-X.

[86] Heineman, G. T. и др. (ред.). *Component-based Software Engineering: Putting the Pieces Together*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. ISBN 0-201-70485-4.

[87] Gamma, E. и др. *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-63361-2.
[88] Keith, P. и др. *Cairo 2D Graphics Library* [онлайн]. [прегл. 2015-05-15]. ⟨`http://cairographics.org`⟩.
[89] Shreiner, D. и др. *OpenGL(R) Library*. 4th. 2007. ISBN 0321514327, 9780321514325.
[90] Vasilev V. Vasilev M., P. A. SolidV – Visual Modeling Framework. В. *SolidV – Visual Modeling Framework*. 2013.
[91] *SolidV Model Doxygen Documentation*. [онлайн]. [прегл. 2015-06-01]. ⟨`http://doxygen.solidopt.org/html/solidv-html/a00031.html`⟩.
[92] *SolidV View Doxygen Documentation*. [онлайн]. [прегл. 2015-06-01]. ⟨`http://doxygen.solidopt.org/html/solidv-html/a00036.html`⟩.
[93] *SolidV Controller Doxygen Documentation*. [онлайн]. [прегл. 2015-06-01]. ⟨`http://doxygen.solidopt.org/html/solidv-html/a00029.html`⟩.
[94] Xamarin. *MonoDevelop IDE* [онлайн]. [прегл. 2015-05-17]. ⟨`http://www.monodevelop.com`⟩.
[95] Association, E. C. M. и др. *Standard ECMA-334: C# Language Specification*. 2005.
[96] Bradley, D. J. *Assembly Language Programming for the IBM Personal Computer*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1984. ISBN 0130491896.
[97] LLVM. *Clang: a C language family frontend for LLVM* [онлайн]. [прегл. 2015-06-02]. ⟨`http://clang.llvm.org`⟩.
[98] Lattner, C. и др. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. В. *Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization*. Palo Alto, California: IEEE Computer Society, 2004, с. 75–. CGO '04. ISBN 0-7695-2102-9.
[99] Torgersen, M. Querying in C#: How Language Integrated Query (LINQ) Works. В. *Companion to the 22Nd ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications Companion*. Montreal, Quebec, Canada: ACM, 2007, с. 852–853. OOPSLA '07. ISBN 978-1-59593-865-7.
[100] Vassilev, V. *ROOT 6 examples with cpplinq* [онлайн]. [прегл. 2015-06-01]. ⟨`https://github.com/vgvassilev/RxCpp/tree/master/Ix/CPP/samples`⟩.
[101] Vasilev, V. и др. Cling–The New Interactive Interpreter for ROOT 6. В. *Journal of Physics: Conference Series*. № 5, 2012, с. 052071.
[102] Chacon, S. и др. *Pro Git*. 2nd. Berkely, CA, USA: Apress, 2014. ISBN 1484200772, 9781484200773.
[103] Dabbish, L. и др. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. В. *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*. Seattle, Washington, USA: ACM, 2012, с. 1277–1286. CSCW '12. ISBN 978-1-4503-1086-4.
[104] Heesch, D. van. *Doxygen* [онлайн]. [прегл. 2015-06-01]. ⟨`http://www.stack.nl/~dimitri/doxygen/`⟩.
[105] Lieberman, H. Programming by Example (Introduction). *Commun. ACM*. 2000, т. 43, № 3, с. 72–74. ISSN 0001-0782.
[106] St. Amant, R. и др. Programming by Example: Visual Generalization in Programming by Example. *Commun. ACM*. 2000, т. 43, № 3, с. 107–114. ISSN 0001-0782.
[107] Петрова, П. *DataMorphose – Визуален анализ и трансформации на данни*. Дипломна работа. Пловдив, 2013. 60 л.
[108] Vassilev, V. и др. DataMorphose: An Interactive Data Migration Framework. В. *DataMorphose: An Interactive Data Migration Framework*. 2014, с. 59–68. ISBN 0-9545660-2-5.
[109] Лесев, Х. *Модулни разширяеми системи за глобално осветяване*. Дисертация. Пловдив, 2015. 160 л.
[110] Лесев, Х. Модулни разширяеми системи за глобално осветяване. Автореферат. Пловдив, 2015. 32 л.
[111] Lesev, H. и др. A Framework for Visual Dynamic Analysis of Ray Tracing Algorithms. *Cybernetics and Information Technologies*. 2014, № 14, с. 38–49. ISSN 1314-4081.
[112] Martin, J. *Rapid Application Development*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc., 1991. ISBN 0-02-376775-8.
[113] Rademakers, F. и др. ROOT: An Object-oriented Data Analysis Framework. *Linux J.* 1998, т. 1998, № 51. ISSN 1075-3583.
[114] CERN. *The Large Hadron Collider* [онлайн]. [прегл. 2015-05-20]. ⟨`http://home.web.cern.ch/topics/large-hadron-collider`⟩.