|  |
| --- |
| Тема на есето |
|
| Фак. номер: 80703 |
| *Христо Атанасов Аврамов* |
| *Компютърни науки, курс 3, поток първи, група 3* |
| *Софтуерни технологии, 2014 г.* |

Съдържание

[1 Увод 3](#_Toc392345817)

[2 Характеристики на Домейн специфичните езици 4](#_Toc392345818)

[3 Противопоставяне на Домейн специфичните езици и езиците с обща цел. 4](#_Toc392345819)

[4 Предимства и недостатаци при употребата на домейн специфични езици. 5](#_Toc392345820)

[5 Предимства и недостатаци при разработка на домейн специфични езици. 7](#_Toc392345821)

[6 7](#_Toc392345822)

[7 Форматиране на таблици и фигури 7](#_Toc392345823)

[7.1 Форматиране и подреждане на фигури 8](#_Toc392345824)

[7.1.1 Под под раздел 8](#_Toc392345825)

[7.1.2 Под под раздел 9](#_Toc392345826)

[7.2 Предаване на есето 9](#_Toc392345827)

[7.2.1 Под под раздел 10](#_Toc392345828)

[7.2.2 Под под раздел 10](#_Toc392345829)

[8 Персонално мнение на автора 11](#_Toc392345830)

[9 Използване на библиографията 11](#_Toc392345831)

[Библиография 12](#_Toc392345832)

# Увод

”Programs must be written for people to read, and only incidentally for machines to execute.”- Този надпис бе избран за девиз на HackFMI v2.0 и бе изписан на тенискитите на всеки от участниците в проявата.Това е един само един от многото цитати , който показват че найстина работата на програмиста върви много по–лесно и по-ползотворно, когато кода с който се работи е интуитивен и близък до човешките взъприятия.Затова термините и понятията във програмните езици трябва да бъдат близки до тези на човека.

Има два типа езици за програмиране които се използват най-често в практиката – GPLs(Езици с обща цел) и DPLs(Домейн специфични езици). Няма точна дефиниция з GPls. Това са езици ,който са предназначени за създаване на приложения с всякакъв вид и насоченост. Тези езици са мощни и ни предоставят богат набор от развити стандарти , множесвто опитни спецялисти владееши ги и голям набор от програми за разработка предоставящи много удобства. От другра страна синтаксиса на GPl езиците е по близък до машинните настройки отколкото до начинът на мислене на човека и това ги прави по-трудни за четене и разбира се писане.

Като дефиниция на DPLs(Домейн специфични езици) може да се приеме следнотоизречение: “A domain-specific language is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain.” Тъй като тези езици решават проблеми от конкретни области , е много по лесно те да навлязат на вътре в концепциите на домейна и да пресъздадат основните му конструкции.Затова и тези езици са по-абстрактни и саответно по близки до мисленето на човека.Примерни за такива езици са DOT за рисуване на графи и SQL за работа с бази от данни , както и много дурги.

Тук ще разгледаме всички аспекти на домейн специфичните езици , като ги съпоставяме със езиците с обща цел(GPls).

# Характеристики на Домейн специфичните езици

Благодарение на проучвания на домейн специфичните езици направени през последните десет годни , успешно са се сформирали, множество характеристики , които са общи за тези езици.Въпреки че всеки от тези езици са специфични всички от тях притежават някои общи качества, който ще опишем в този раздел.

Обикновенно DLS–ите са малки езици,защото с тях се цели да се разрешат контректни проблеми , такаче операциите и структорите , който ще се реализират посредством езика са ограничени.Заради малкия си размер тези езици са повече делкаративни отколкото императивни.

Друга характеристика на Домейн специфичните езици е, че те са абстрактни .Отново поради, това че обслужват конкретен домейн е по лесно да се пресъздадат концепциите , така че да бъдат близки до човешкото мислене и така хората които ги използват успявят много по-лесно да направят между структурите на езика и тези на домейна.

Горните две характеристики на езика позволяват да бъдат на тези езици да бъдат писани и четени от домейн специалисти.Това са хора, които са с дълбоки и разширени познания по дадения домейн, но обикновенно със слаби или никави такива в обласата на програмирането.Малкия размер на домейн специфичните езици и тяхната абстрактивност позовляват на домейн експертите лесно да навлязат в същноста им и с минимални усилия да започнат да ги използват

# Противопоставяне на Домейн специфичните езици и езиците с обща цел.

След основните характеристики на Домейн специфичните езици посочени по горе се повдига въпроса за разликите между тях и езиците с обща цел.Основните противопоставяния на тези видове езици ще направим в три категории.

Първото такова е абстракцията на домейн специфичните срещу конкретиката на тези за общя цел.Един език е толкова по абстрактен, колкото повече скрива семантиката на операциите си. При DSL-ите това е силно застъпено, когато се постига интутивност и добливажане до човешките възприя, затова без колебание можем да кажем че домейн специфичните езици са по абстрактни от тези с обща цел.Разбира се , с повечето абстрактност идват плюсове и минуси, но несъмнено плюсовете надделяват.

Друго такова противопоставяне е противопоставянето им като езици от жисоко и от ниско ниво.Това сравнение естествено е пряко свързано с гроното . Благодарение на това че DSL-ите са по абстрактни от GPL –ите и се са по интуитивни за използващите ги можем да кажем че те са езици от високо ниво в това сравнение.А и философията на използване на езиците от виското ниво е точно потребителите да се концентрират върху проблема а не върху имплементирането на решението.

Треторо такова провопоставяне ще бъде в посока изразителност срещу изчислителна мощ.Един език е изразителен, ако използващия го чрез кода и съпътствящите го коментари(без необходимост от четене на допълнителна документация) може да се ориентира и лесно да чете , а в последствие да пише на този език.Изчислителната мощ на език пък е свързана с възможността за използване на множество и различни операции и изчисления чрез структорите на езика.Ясно е , че колкото по малък е един език толкова по изразителен е , но за сметка на това губи от своята изчислителна мощ. За това казваме, че домейн специфичните езици са по-изразителни, а езиците с обща цел с по голяма изразителна мощ.Това е и основната разлика между тези два вида езици.

# Предимства и недостатаци при употребата на домейн специфични езици.

За най-голямото предимство на домейн специфичните езици се смята възможността за лесно интегриране на домейн специалисти.Обиновенно работата на тези експерти свършва с анализирането и изясняването на концепциите на проблема.Но докато при езиците за общи цели са необходими дълбоки знания в областа на програмирането ,тук дори е възможно да се използват домейн експертите във фазата на разработка.Този факт позволява разработката на множество нови методологии за разработка на софтуер ,пригодени към нуждите на конкретния домейн и носещи видимо по добри резултати.

Както вече споменахме използването на DSl и може значително да намали времето за разработка на софтуер.Това се дължи на едни от основните предимства на тези езици, а именно : гъвкавост, производителност и изполваемост .Истинноста на тези качества е пряко следсвтие от основните характеристики на домейн специфичните езици.

Домейн специфичните езици също намаляват времето и разходите за фазата на тестване , и подръжката на разработените с тях софтуери .

Пряко следствие от изразителността на тези езици е и друг важен техен плюс – самия код представлява своеобразна документация. Това спомага изключително много процеса на разработка и рефакториране на кода,като намалява не само времето но и разходите в тези фази. Също така при домейн специфичните езици, запознаването на нови кадри с проекта , и изискванията по него става много по-бързо и лесно.

Дотук посочихме само позитивните страни на домейн специфичните езици но естествено, като всичко на този свят, те си имат и негативи. Първото такова е, че те са тясно специализирани, и обикновенно се налага човек да знае по няколко DSL езика ,и въпреки това е много вероятно при започване работя по нов проект , да му се наложи да учи нов език.За това много програмисти казват, че предпочитат да знаят един език за общи цели, вместно много специфични. Въпреки това се смята ,че домейн специфичните езици са лесни за научаване поради малкия им обем и изразителността им.

Друг недостатък е , че в реални проекти , в който има повече от един домейн , често се изполсват повече от един DSL , а понякога дори се среща комбинацията на GPL с DSL ит.Ясно е,че при недобри механизми за комбиниране на тези езици , това може да се превърне в един сериозен проблем.

# Предимства и недостатаци при разработка на домейн специфични езици.

# 

# Форматиране на таблици и фигури

Ако използвате таблици по-долу е даден вида и формата на примерна таблица. Таблиците се именуват със центриран текст под тях, който съдържа кратко описание на предназначението на таблицата.

Пример:

| Изисквания | Описание |
| --- | --- |
| **Работоспособност** |  |
| **Сигурност** |  |
| **Използваемост** |  |
| **Тестируемост** |  |
| **Еволюция** |  |
| **Адаптация и самоадаптация** |  |

**Таб. 1**. *Таблицата представлява пример за оформяне на таблица*

Номерацията на таблиците се прави последователно в нарастващ ред на номерата като се започне от 1.

## Форматиране и подреждане на фигури

Използването на фигури, диаграми и схеми изисква те да бъдат номерирани и именувани по следния начин:



**Фиг. 1**. *Фигура, демонстрираща номерацията и формата*

Номерацията на фигурите се прави последователно в нарастващ ред на номерата като се започне от 1.

Фигурите се центрират в текстта. Около тях не се поставя текст. Текстта предшества или следва фигурата.

***Стила в който се форматират подразделите е Heading2***.

### Под под раздел

Това е текст формиран в цялостен параграф ***Стила в който се форматират подподразделите е Heading3***.

Това е текст формиран в цялостен параграф в няколко изречения Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф ***Стила в който се форматират подподподразделите е Heading4.*** Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

### Под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

## Предаване на есето

Есето се предава в електронен вид след щателно оглед за спазване на формата на текстта съгласно изикванията на този темплейт по електронната поща. След проверка и коментар от страна на проверяващия есето се коригира, допълва или се разработват допълнително негови отделни елементи. След което се изпраща отново по електронната поща в екетронен формат на Microsoft Word.

### Под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

### Под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

#### Под под под раздел

Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф Това е текст формиран в цялостен параграф

# Персонално мнение на автора

Едно от изискванията към есето е то да съдържа раздел с описание на личното мнение и отношение на автора към дискутираната тематика. Възможно е есето да не съдържа специален раздел за целта. В този случай мнението на автора трябва да е видно и ясно изразено в останалите части от текстта.

# Използване на библиографията

Дадената по-долу библиография служи за пояснения на термини, названия, методи, и други елементи, които са взаимствани от съответната препратка или пък дават по-обширно пояснение на означения текст. Препраткит екъм библиографията изглеждат така: „Garlan дефинира пътна карта [Gar00] на софтуерните архитектури. По-късно той дефинира понятието архитектурен хаос [GAO09]. ...“ или „програмната евол\ция , описана от M. Lehman и L. Belady [LB85] позволява ....“

Използването на препракти от интернет или цпрепратки към страници и конкретни сайтове се извършва по подобен начин: „ Средата за разработка Eclipse [ECLPS] на разработчиците да редактират и компилират код на езика Java […]. В този случай е достатъчно еднократно цитиране на инструмента или термина еднократно при първото негово споменаване в текстта.

# Библиография

Запова на нова страница.

Библиографията се формира по следния начин – цитиран в начупени скоби [ ] се поставя следния текст:

[Asi02] Nadim Asif, “Architecture Recovery”, In the Proc. of International Conference of Information and Knowledge Engineering (IKE’02), lasVegas, 2002.

[Gar00] D. Garlan, “Software Architecture: A Roadmap,” Proc. Int’l Conf. Software Eng., pp. 91-101, 2000.

Ако автора е един се вземат първите три букви н фамилията след това се слагат двете последни цифри от годината на излизане на публикацията

Ако авторите са двам или трима се взимат първите им букви на фамилиите и се поставят последните две цифри от годината на излизане на публикацията

[LB85] M. Lehman and L. Belady, Program Evolution: Processes of Software Change. London Academic Press, 1985.

[JRL00] M. Jazayeri, A, Ran, F. van der Linden, Software Architecture for Product Families Principles and Practice, Addison Wesley, 2000.

[GAO09] D. Garlan, R. Allen, and J. Ockerbloom, “Architectural Mismatch: Why Reuse Is So Hard,” IEEE Software, vol. 26, no. 4, pp. 66-69, Nov. 2009.

Ако авторите са повече от трима се взимат първите букви от фамилните имена на първите двама автори, след това се поставя + и след това последните две цифриот годината на издаване н апубликацията

[KM+06] J. Knodel, D. Muthig, M. Naab, and M. Lindvall, “Static Evaluation of Software Architectures,” Proc. Conf. Software Maintenance and Reeng., pp. 279-294, 2006.

[GP+09a] Joshua Garcia , Daniel Popescu , George Edwards , Nenad Medvidovic, Identifying Architectural Bad Smells, Proceedings of the 2009 European Conference on Software Maintenance and Reengineering, p.255-258, March 24-27, 2009 [doi>10.1109/CSMR.2009.59]

[GP+09b] Joshua Garcia, Daniel Popescu, George Edwards and Nenad Medvidovic. Toward a Catalogue of Architectural Bad Smells. Proceedings of the Fifth International Conference on the Quality of Software Architectures (QoSA 2009), East Stroudsburg, Pennsylvania, USA, June 2009.

Ако две публикации с един и същи автори в една и съща година има няколко две три и повече след цифрите на годината се добавя последователно a,b,c и т.н. както по-горе

Примери:

[Hol01] R. Holt, “Sofware Architecture as a Shared Mental Model,” Proc. ASERC Workshop Software Architecture, 2001.

[MEG03] N. Medvidovic, A. Egyed, and P. Gruenbacher, “Stemming Architectural Erosion by Architectural Discovery and Recovery,” Proc. Int’l Workshop from Software Requirements to Architectures, 2003.

[MMP00] N. R. Mehta, N. Medvidovic, and S. Phadke. Towards a taxonomy of software connectors. In Proc. of the 22nd Int. Conf. on Software Engineering, 2000.

[PW92] D.E. Perry and A.L. Wolf, “Foundations for the Study of Software Architecture,” ACM SIGSOFT Software Eng. Notes, vol. 17, no. 4, pp. 40-52, 1992.

[Riv04] C. Riva, “View-Based Software Architecture Reconstruction,” PhD thesis, Technical Univ. of Vienna, 2004.

[TMD08] R. Taylor, N. Medvidovic, and E. Dashofy. Software Architecture: Foundations, Theory, and Practice. John Wiley & Sons, 2008.

[WPM10] David Woollard, Chris A. Mattmann, Daniel Popescu, and Nenad Medvidovic. KADRE: Domain-specific architectural recovery for scientific software systems, 25th IEEE/ACM International Conference on Automated Software Engineering, 2010.

[MK96] N.C. Mendonc¸a and J. Kramer, “Requirements for an Effective Architecture Recovery Framework,” Proc. Int’l Software Architecture Workshop, pp. 101-105, 1996.

[OSV02] L. O’Brien, C. Stoermer, and C. Verhoef, “Software Architecture Reconstruction: Practice Needs and Current Approaches,” Technical Report CMU/SEI-2002-TR-024, Carnegie Mellon Univ., 2002.

[DP09] Stephane Ducasse, Damien Pollet, "Software Architecture Reconstruction: A Process-Oriented Taxonomy," IEEE Transactions on Software Engineering, vol. 35, no. 4, pp. 573-591, July/August, 2009

[Deu02] Arie van Deursen, Software architecture recovery and modeling: [WCRE 2001 discussion forum report], ACM SIGAPP Applied Computing Review, v.10 n.1, p.4-7, Spring 2002 [doi>10.1145/568235.568236]

[RA07] Ghulam Rasool, Nadim Asif, Software Architecture Recovery, World Academy of Science, Engineering and Technology 4 2007

Цитиране на ресурси от интернет страници става с помощта на съкратено наименование до 5 символа

[ECLPS] https://www.eclipse.org/ (последно използвана на 04.2014)

Текстта на линка запазете в черен цвят – изтрийте по възможност хиперлинка на връзката. В кръглите скоби поставете датата на последен достъп на връзката от ваша страна. Връзката след време може и да не е валидна или преместена.