|  |  |  |
| --- | --- | --- |
|  | **Софийски университет „Св. Кл. Охридски”**  Факултет по математика и информатика  *Бакалавърска програма „Софтуерно инженерство”* |  |

**Предмет: XML технологии за семантичен Уеб**

***Зимен семестър, 2021/2022 год.***

**Тема 63: Преглед на Graph Markup Language (GraphML)**

**Есе**

*Автори:*

*Мирослав Дионисиев, фак. номер 62390*

*Павел Сарлов, фак. номер 62393*

декември, 2021

София

Съдържание

[1 Въведение 3](#_Toc496793431)

[2 Характеристики и използване на езика/ средата/ технологията/ … 3](#_Toc496793432)

[2.1 Дефиниции 3](#_Toc496793433)

[2.2 Основни характеристики 4](#_Toc496793434)

[2.2.1 Характеристика 1 4](#_Toc496793435)

[2.2.2 Характеристика 2 4](#_Toc496793436)

[2.3 Въведение в използването на езика/ средата/ технологията/ … 5](#_Toc496793437)

[2.4 Ограничения при използването на езика/ средата/ технологията/ … 5](#_Toc496793438)

[2.5 Други 5](#_Toc496793439)

[3 Сравнителен анализ 5](#_Toc496793440)

[3.1 Критерии за сравнение 5](#_Toc496793441)

[3.2 Сравнение с език/среда/технология/… 1 6](#_Toc496793442)

[3.3 Сравнение с език/среда/технология/… 2 6](#_Toc496793443)

[4 Примери на използване 6](#_Toc496793444)

[4.1 Пример 1 7](#_Toc496793445)

[4.2 Пример 2 7](#_Toc496793446)

[5 Добри практики и методи за използване 7](#_Toc496793447)

[6 Заключение и очаквано бъдещо развитие 8](#_Toc496793448)

[7 Разпределение на работата 8](#_Toc496793449)

[8 Използвани литературни източници 8](#_Toc496793450)

# Въведение

Науката, която се занимава с изучаване на графите и задачите свързани с тях, се нарича „Теория на графите“. Началото ѝ е поставено от математика Леонард Ойлер, който решава задачата за седемте моста на Кьонигсберг, която търси маршрут от квартал в града, минава през всичките седем моста точно по веднъж и завършва в същия квартал.

В практиката множеството от софтуерни системи използват графите като средство за решаване на разнообразни по своята същност проблеми. Примери за това са приложенията за навигация като Google Maps, GPS системите и други. Графи се използват също така и при съхранението на данни, защото улесняват съхраняването както на структурирани, така и на неструктурирани такива.

Системите, използващи данни под формата на графи, имат нужда от формат, под който да съхраняват и обменят тази информация. Разбира се, при различните системи тези дейности се извършват по различен начин и през годините са правени много опити за унифицирането на комуникацията помежду им.

Мотивирани от целите за оперативна съвместимост на инструментите, достъп до набори от сравнителни данни и обмяна на данни в мрежата, Управителният комитет на Симпозиума по изобразяване на графи стартира нова инициатива с неформален семинар, проведен във връзка с 8-мия Симпозиум по изобразяване на графи (GD 2000). В резултат на това се сформира неформална работна група, която да предложи модерен формат за обмен на данни между инструменти за изобразяване на графи и други приложения. Този формат бива наречен Graph Markup Language (GraphML), който благодарение на XML-базирания си синтаксис, може да се използва в комбинация с други XML-базирани формати. [1]

# Характеристики и използване на GraphML

## Дефиниции

* Граф – представлява абстрактна структура, имаща за цел да представи данните в свързан вид. В математиката графът *G* се представя чрез наредената тройка *G=(V, E, D)*, където *V* e множество от възли, *E* е множество от ребра, а *D* е функция, съпоставяща на всяко ребро двойка върхове.
* Възел – елемент на графа, в който се съдържа основната негова информация. Всеки възел може да съдържа данни.
* Ребро – представлява връзката между възлите. Ребрата биват насочени и ненасочени. Всяко ребро може да притежава тегло, чрез което да се сравни с другите ребра.

## Основни характеристики

Синтаксисът на GraphML е описан в GraphML Schema[[1]](#footnote-1). Въпреки че това е основната дефиниция, също така се използва и по-слаба спецификация в GraphML DTD, която не различава референтни типове като идентификатори на възли и ребра. Въпреки това, някои приложения се нуждаят от DTD, за да работят коректно.[[2]](#footnote-2)

### Разширяемост

В GraphML, освен стандартните данни, може да се съхранява още разнообразна по вид и формат информация. Това се постига чрез промени в дефиницията на GraphML схемата, като се добавят XML тагове и атрибути. По този начин може да се предостави допълнителна информация на приложения, които я изискват, или дааа се съставят по-сложни екземпляри.

### Опростеност

Форматът GraphML е създаден с цел лесно да се анализира и интерпретира както от хора, така и от машини. Не съществуват неясноти – всеки валиден GraphML документ има своя единствена интерпретация.

### Генерализация

Няма ограничения относно теорията на графите – т.е. йерархични графи, хиперграфи и прочие могат да бъдат представяни чрез основния синтаксис на езика, без да е нужно да се добавят разширения към синтаксиса.

### Устойчивост

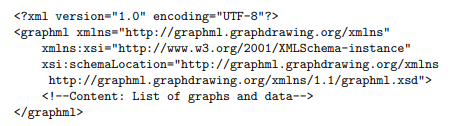
Успоредно със своята разширяемост, GraphML предоставя и прозрачност на добавената информация. Това означава, че системи, които не могат да обработват всички видове графи или добавената към тях информация, могат лесно да извличат единствено тази, която им е нужна.

### XML синтаксис

Благодарение на своя XML синтаксис, GraphML може да се използва в комбинация с други XML-базирани формати. Тъй като изобразяването на GraphML графи често се нуждае от предварителна обработка или трансформация до други XML формати, изключително удобно е да се използва XSLT, език специално създаден за трансформацията на XML документи.

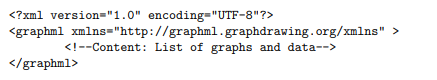
## Въведение в използването на GraphML

### Заглавна част

За да бъде валиден, всеки XML документ трябва да започва с декларация на DTD или XML Schema. Езикът на GraphML е дефиниран чрез гореспоменатата схема. На фиг. 1 е показан минимума, изискван от един GraphML документ.

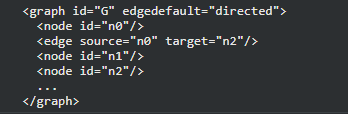
Фигура 1: минимален валиден GraphML документ [1]

Пъврият ред от документа на фиг. 1 е процесорна XML инструкция, която дефинира стандарта (XML 1.0) и кодирането му (UTF-8). Вторият ред съдържа кореновия елемент на един GraphML документ: *graphml*, който подобно на всички други GraphML елементи принадлежи на пространството от имена <http://graphml.graphdrawing.org/xmlns>. Поради тази причина пространството от имена по подразбиране, дефинирано чрез XML атрибута xmlns=”<http://graphml.graphdrawing.org/xmlns>”, е именно това. Следващите два атрибута декларира коя XML Schema е използвана за валидация на документа. Атрибутът xmlns:xsi=“<http://www.w3.org/2001/XMLSchema-instance>“ дефинира xsi като префиксно пространство от имена за схемата, а атрибутът xsi:schemaLocation=“<http://graphml.graphdrawing.org/xmlns> <http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd>“ дефинира локацията на XML схемата за пространството от имена на GraphML.

Валидацията на такъв вид документи не е задължителна. При липса на необходимост, тя може да бъде пропусната. Един минимален GraphML документ без нея би изглежда по следния начин:

Фигура 2: минимален GraphML документ без референция към схема [1]

### Самият граф

Един граф се определя от *graph* елемент. В него се влагат декларациите на възлите и ребрата. Възлите се декларират посредством *node* елемента, а ребрата - *edge* елемента. В GraphML няма определен ред на дефиниране на възли и ребра. Следователно следния пример е напълно валиден GraphML фрагмент:

Фигура 3: граф с неопределен ред на дефинирани ребра и възли [2]

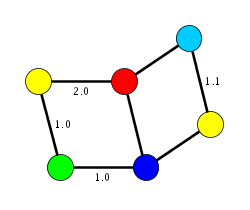
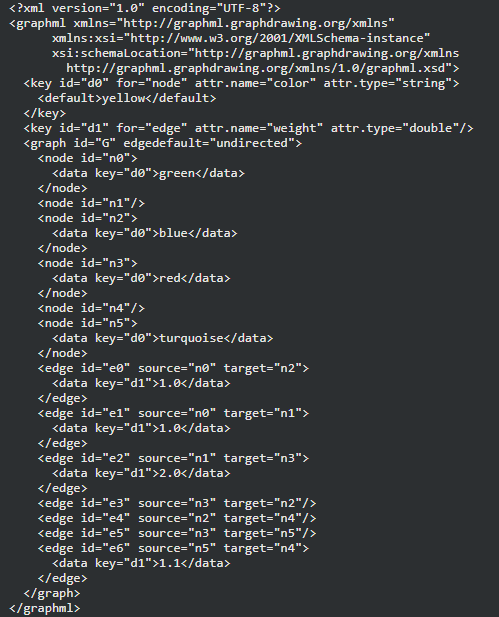
* Графите в GraphML са смесени, т.е. съдържат едновременно насочени и ненасочени ребра. Ако няма определена посока на реброто, тя се определя от посоката по подразбиране, която се декларира чрез XML атрибута *edgedefault* на *graph* елемента. Двете възможни стойности на този атрибут са *directed* и *undirected*. Добре е да се отбележи, че посоката по подразбиране трябва да се специфицира.
* Възлите на графа се декларират чрез *node* елемента, като всеки възел си има идентификатор, който трябва да бъде уникален в целия документ, т.е. в един документ не трябва да има два възела със съвпадащи идентификатори. Идентификаторът на един възел се дефинира чрез XML атрибута *id.*
* Ребрата на графа се декларират чрез *edge* елемента. Всяко ребро трябва да дефинира двете си крайни точки чрез XML атрибутите *source* и *target*. Стойността на *source*, съответно на *target,* трябва да бъде идентификатор на възел от същия документ. Ребра само с една крайна точка, също наречени цикли, се дефинират чрез еднакви стойности за *source* и *target.* Незадължителният атрибут *directed* определя дали реброто е насочено или не. Стойността *true* декларира насочено ребро, обратно *false* – ненасочено ребро. Ако посоката не е строго дефинирана се използва посоката по подразбиране за съответното ребро, дефинирана в обхващащия го граф (Фиг. 4).

Фигура 4: дефиниция на ребро с всички XML атрибути [2]

### Атрибути

В предишната секция разгледахме как се описва топологията на граф в GraphML. Въпреки че чисто топологична информация е достатъчна за някои приложения на GraphML, в повечето случаи е нужда и допълнителна информация. С помощта на разширението GraphML-Attributes може да се специфицира допълнителна информация от прост тип (числови стойности или низове) за елементите на графа.

Ако се налага да се добави структурирано съдържание към елементите на графа, трябва да се използва разширения механизъм на GraphML за ключ/данни. Самите GraphML атрибути са специализирани такива разширения. GraphML атрибутите не бива да се бъркат с XML атрибутите, които са напълно различна концепция.

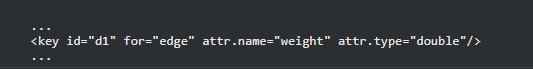
Нека разгледаме следния пример на граф с оцветени възли и претеглени ребра:

Фигура 6: GraphML документ с GraphML атрибути [2]

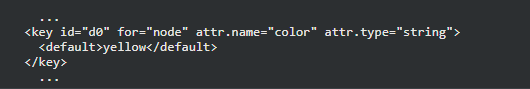
Фигура 5: граф с оцветени възли и претеглени върхове [2]

Ще използваме GraphML атрибутите за съхранението на допълнителните данни за възлите и ребрата. Можем да видим резултата на фиг. 6.

GraphML атрибутите се дефинират чрез *key* елемент, който определя идентификатор, име, тип и домейн на атрибута.

* Идентификаторът се специфицира от XML атрибута *id* и се използва за реферирането към GraphML атрибут в документа.
* Името на GraphML атрибута се определя от XML атрибута *attr.name* и трябва да бъде уникално сред всички декларирани GraphML атрибути в документа. Целта на името е да позволи на приложенията да идентифицират значението на атрибута. Хубаво е да се отбележи, че името на GraphML атрибута не се използва вътре в самия документ, за тази цел се използва идентификаторът.
* Типът на GraphML атрибутите може да бъде един от *boolean, int, long, float, double* или *string*. Тези типове са дефинирани подобно на типовете в Java програмния език.
* **Домейнът на GraphML атрибутите определя за кои елементи от графа е деклариран самия атрибут. Възможни стойности включват *graph, node, edge* и *all.*

Фигура 7: декларация на GraphML атрибут [2]

Възможно е също така да се дефинираа и стойност по подразбиране на GraphML атрибут. Това се определя от текстовото съдържание на *default* елемента.

Фигура 8: декларация на GraphML атрибут със стойност по подразбиране [2]

Стойностите на GraphML атрибут за елемент от графа се дефинират чрез елемента *data*, който се влага в съответния елемент от графа. За *data* се използва XML атрибута *key*, който реферира към идентификатора на GraphML атрибута. Стойността на GraphML атрибута е текстовото съдържание на *data* елемента. Тази стойност трябва да бъде със същия тип, деклариран в съответната *key* дефиниция.

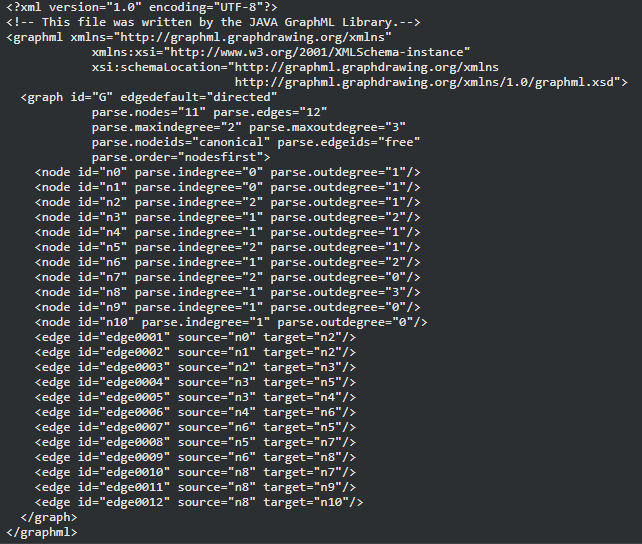
Фигура 9: стойност на GraphML атрибут [2]

Може да има елементи от графа, за който е дефиниран GraphML атрибут, но няма декларирана стойност чрез съответен *data* елемент. Ако име дефинирана стойност по подразбиране за този GraphML атрибут, то се използва тя. В примера на фиг. 9 няма дефинирана стойност за възела с идентификатор *n1* и GraphML атрибутът с име *color*. Следователно този GraphML атрибут има стойност по подразбиране *yellow* за текущия възел. Ако няма специфицирана стойност по подразбиране, както при GraphML атрибута *weight*, стойността на съответния атрибут не е дефинирана за елемента на графа. В примера от фиг. 9 стойността на GraphML атрибута *weight* за реброто с идентификатор *e3* не е дефинирана.

### Информация за парсера

С цел да се направи възможна имплементацията на оптимизирани парсери на GraphML документи може да се прикрепят метаданни към XML атрибутите на някои GraphML елементи. Всички XML атрибути, обозначаващи метаданни, се предхождат от *parse*. Съществуват два вида метаданни:

* информация за броя на елементите – дефинирани са следните XML атрибути:
  + за *graph* елемента:
    - *parse.nodes* – обозначава броя на възлите в графа;
    - *parse.edges* – обозначава броя на ребрата в графа;
    - *parse.maxindegree* – обозначава максималния брой на входящите ребра;
    - *parse.maxoutdegree –* обозначава максималния брой на изходящите ребра;
  + за *node* елемента:
    - *parse.indegree* – обозначава броя на входящите ребра;
    - *parse.outdegree* – обозначава броя на изходящите ребра;
* информация за кодировката на данните в документа – дефинирани са следните XML атрибути:
  + за *graph* елемента:
    - *parse.nodeids* – има стойност *canonical*, всички възли имат идентификатори следващи шаблона *nX,* където *X* обозначава броя на срещанията на предходния *node* елемент; в противен случай стойността на атрибута е *free*;
    - *parse.edgeids* – аналогично на *parse.nodeids* с единствената разлика, че идентификаторите на ребрата следват шаблона *eX*;
    - *parse.order* – обозначава подредбата на *node* и *edge* елементите в документа; при стойност *nodesfirst* не е позволено на *node* елемент да присъства след появата на първия *edge* елемент; при стойност *adjacencylist*, декларацията на възел се следва от декларацията на съседните му ребра; при стойност *free* няма строга подредба;

На фиг. 10 се демонстрира информацията от метаданни за парсера от нашия пример:

Фигура 10: граф с допълнителна метаданни за парсера [2]

## Ограничения при използването на GraphML

… Насоки за разработка на главата:

* Тук опишете ограниченията при използване на езика/средата/технологията/…;
* Дайте примери

## Други

…

Можете да включите и други секции според темата и предпочитанията ви, като напр. семантика, динамика, правила, и др.

# Сравнителен анализ

Тук представете сравнителен анализ на езика/ средата/ технологията/ … Избройте с какви други подобни на езици/ среди/ технологии/ … ще извършите сравнението.

Най-напред опишете критериите, по които извършвате сравнението.

## Критерии за сравнение

…

Опишете защо използвате точно тези критерии и техните особености и значимост.

## Сравнение с език/среда/технология/… 1

… Можете да използвате и сравнение в табличен вид.

Таблица 1: Сравнение на … с ….

|  |  |  |
| --- | --- | --- |
| **Критерий** | **Разглеждан език/среда/технология/…** | **Език/среда/технология/… 1** |
| Критерии 1 | …. | …. |
| Критерии 2 | …. | …. |
| Критерии 3 | …. | …. |
| Критерии 4 | …. | …. |

## Сравнение с език/среда/технология/… 2

…. Можете да използвате и сравнение в табличен вид, като таблици 1 и 2 бъдат слети (т.е. представите всички разглеждани подобни езици/среди/… заедно в една обща таблица).

Таблица 2: Сравнение на … с ….

|  |  |  |
| --- | --- | --- |
| **Критерий** | **Разглеждан език/среда/технология/…** | **Език/среда/технология/… 2** |
| Критерии 1 | …. | …. |
| Критерии 2 | …. | …. |
| Критерии 3 | …. | …. |
| Критерии 4 | …. | …. |

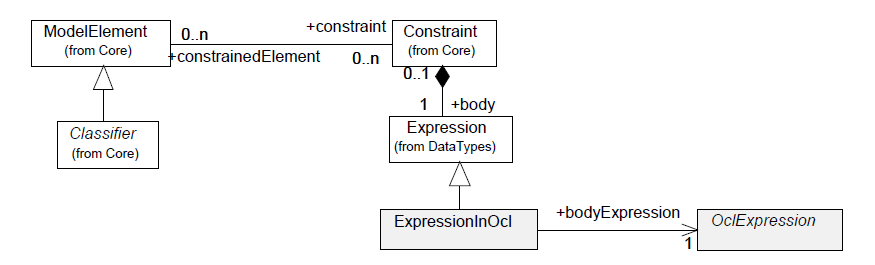
# Примери на използване

….

Представете и коментирайте примери на използване на езика/ средата/ технологията/ …

## Пример 1

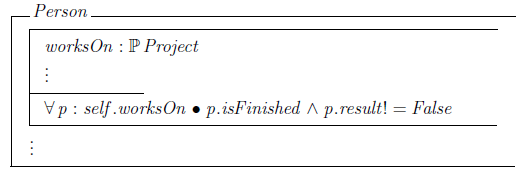
….



Фигура 2. Ситуация, при която OCL изразът е използван като дефиниция или инвариант фигура (източник: [3]).

## Пример 2

….



Фигура 3. ………………. (източник: [4]).

# Добри практики и методи за използване

Тук представете добри практики и методи/начини на използване на езика/ средата/ технологията/ …

…

# Критични бележки и авторска позиция

Представете Вашата оценка за анализирания език, технология, продукт… - как критично го преценявате Вие с оглед на практиката и развитието на ИТ.

…

# Заключение и очаквано бъдещо развитие

Обобщете значимостта на предмета на есето, използваемостта му в практиката и предлаганите към настоящия момент насоки за бъдещо развитие…

…

*Заб.: есето се предава само в електронен вид (в MS Word или PDF формат), чрез качването на файла с документа в задание за предаване на есе, в страницата на курса в Moodle.*

# Разпределение на работата

Опишете как е била разпределена работата между студентите, работили по есето (в случай на повече от един автори).

# Използвани литературни източници

1. [Graph Markup Language (GraphML)](https://cs.brown.edu/people/rtamassi/gdhandbook/chapters/graphml.pdf)
2. Литературен източник 2
3. Литературен източник 3
4. Литературен източник 4
5. Литературен източник 5

1. [GraphML Schema](http://graphml.graphdrawing.org/xmlns/1.1/graphml.xsd) [↑](#footnote-ref-1)
2. [GraphML Specification (graphdrawing.org)](http://graphml.graphdrawing.org/specification.html) [↑](#footnote-ref-2)