XML валидиране чрез Document Type Definition (DTD)

Цели
Структура на DTD
Синтаксис
Особености
Единици
Примери

XML синтаксис накратко

- Elements
  - XML tags for markup
- Attributes
  - Tuple information of elements
- Declarations
  - Instructions to XML processor
- Processing Instructions
  - Instructions to external applications

Дефиниция на типа на документа (DTD)

- DTD дефиницията задава шаблон за маркиране на XML документ
- Форматът на DTD е наследен от SGML, като значително е опростен
- Както при SGML, така и XML DTD използва формална граматика за описание на структурата и типа на съдържанието на XML документа
- DTD осигурява начин за проверка на неговата структура и съдържание – т.е. за ***валидация***

DTD и XML schema

- DTD (Document Type Definition) и XML Schema или XSD (XML Schema Definition) задават правила, съгласно които се определят имената на елементите и атрибутите, тяхната последователност, честота на срещане и др.
- DTD използва по-стегнат и кратък синтаксис в сравнение с XML Schema, но за сметка на това XML Schema предоставя по-богат набор от средства за по-строго дефиниране на структурата на XML и освен това нейните правила се задават в XML формат.

Валиден XML документ

- Всеки отделен документ, отговарящ на даден документен тип, е документен екземпляр (instance) на типа. Такъв документ представлява валиден документ за този документен тип.
- Всеки валиден документ е добре конструиран, но обратното не е задължително вярно.
- Всички XML парсери проверяват дали входния документ е добре конструиран XML документ.
- Парсери, които извършват още и проверка за определяне дали съдържанието на XML документите е валидно спрямо зададен тип на документа, се наричат валидиращи парсери.

Валидация

- Валидацията е времеемък процес, но често спестява много проблеми на външните приложения и се извършва от специализиран процесор (валидиращ парсер)
- Въпрос: *какви са ползите от валидацията?*
- DTD описанието е споделено от валидизиращия парсер за XML документите-екземпляри (instances) на този документен тип – т.е. използва се многократно само едно описание

Същност на DTD

DTD документите описват правилата за структуриране на документа, изграждащите го елементи, възможните типове атрибути и стойностите им по подразбиране. По-конкретно, те задават:

- *Какви **имена** на елементи и техни атрибути могат да бъдат използвани в документа;*
- *Какви са **типовете** на елементите и техните атрибути;*
- *Каква е **йерархията** на документа;*
- *Какви видове **единици** се ползват в него.*

Начини за използване на DTD

Спрямо валидирания XML документ-екземпляр, DTD документът може да бъде:

- *Вътрешен за* XML *документа* – допуска се най-много един вътрешен DTD документ;
- *Външен за* XML *документа* – използваните външни DTD документи могат да бъдат повече от един;
- *Едновременно използване на един вътрешен и друг/други външен/външни XML документи*

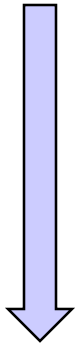
Реферирane към вътрешно DTD

```
<?xml version="1.0" encoding="windows-1251" ?>
<!DOCTYPE  recipe [
        <!ELEMENT  recipe (category,author,title,body)>
        <!ELEMENT category (#PCDATA)>
        <!ELEMENT author   (#PCDATA)>
        <!ELEMENT title    (#PCDATA)>
        <!ELEMENT body     (#PCDATA)>
]>
<recipe>
  <category>Cakes</category>
  <author>Ralica</author>
  <title>Chocolate cake</title>
  <body>Products: ….
        Way of preparation:….
        Result: Very delicious!
   </body>
</recipe>
```

Допустимо е само едно вътрешно DTD описание

Използване на външна DTD дефиниция

- <?xml version="1.0" encoding="windows-1251" ?>
- <!DOCTYPE recipe SYSTEM "example-DTD-2.dtd">
- <recipe>
- <category>Cakes</category>
- <author>Ralica</author>
- <title>Chocolate cake</title>
- <body>Products: ….
- Way of preparation:….
- Result: Very delicious!
- </body>
- </recipe>



```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT                                    recipe
(category,author,title,body)>
        <!ELEMENT category (#PCDATA)>
        <!ELEMENT author   (#PCDATA)>
        <!ELEMENT title    (#PCDATA)>
        <!ELEMENT body     (#PCDATA)>
        <!ENTITY footer "www.recipe.com">
        <!ATTLIST author profession  CDATA
#REQUIRED>
```

Външни DTD дефиниции

- **<?xml version="1.0"?>**
- **<!DOCTYPE Rolodex SYSTEM "rolodex.dtd">**
- **. . .** *{ XML document instance goes here }*

DESCRIPTOR = SYSTEM or PUBLIC :

- **SYSTEM – local file or URL**
- **PUBLIC (for public catalogues) – a catalog entry in Formal Public Identifiers format (FPI)**

Комбинация от двата подхода

• с дефиниран DTD както в отделен документ, така и в самия XML. В този случай вътрешната DTD може да предефинира само **ENTITY** и **ATTLIST** на външната дефиниция. Пример:

```
<?xml version="1.0" encoding="windows-1251" ?>
<!DOCTYPE recipe SYSTEM "example-DTD-2.dtd" [
   <!ENTITY footer "www.myrecipe.com">
   <!ATTLIST author title CDATA #REQUIRED> ]>
   <recipe>
   <category>Cakes</category>
   <author title="eee" profession="musician">   &footer;
   </author>
   <title>Chocolate cake</title>
   <body>Products: ….
                Way of preparation:….
   </body>
   <comment> Very delicious!</comment>
</recipe>
```

DTD декларации

- Инструкции за XML процесора
- Формат - `<! … >` or `<! … [<! … >]>`
  - Document type: `<!DOCTYPE … >`
  - Character data: `<![CDATA[ … ]]>`
  - Entities: `<!ENTITY … >`
  - Notation: `<!NOTATION … >`
  - Element: `<!ELEMENT … >`
  - Attributes: `<!ATTLIST … >`
  - Conditional sect.'s: `<![INCLUDE[…]]>` `<![IGNORE[…]]>`

Дефиниране на елементи в DTD

- Ако елементът съдържа само текст:

<!ELEMENT element_name (#PCDATA)>

- Ако е празен:

<!ELEMENT element_name EMPTY>

- Ако е с елементно съдържание:

<!ELEMENT element_name (child_1, child_2)>

Валиден XML съгласно тази DTD дефиниция е:

```
<element_name>
  <child_1>   .....   </child_1>
  <child_2>   .....   </child_2>
</element_name>
```

Избор на елементно съдържание

Под-елементите могат да бъдат разделени и със знака '|', който има значение на логическия оператор *OR*:

<!ELEMENT element_name
((child_1, child_3) | (child_2, child_4))>

Честота на срещане на елементи

- **?** – със значение нула или веднъж,
- ***** - със значение нула или повече пъти,
- **+** - веднъж или повече пъти.
- Тези *оператори за честота* се поставят след съответния елемент / елементи така:
  - **<!ELEMENT element-name**

    (child_1*, ((child_2, child_3?) | child_4+)*,...)>
- Ако липсва такъв оператор след даден елемент или множество от елементи, то те трябва да се срещнат в съответния XML точно веднъж.

Елементи с произволно съдържание

- **<!ELEMENT element-name ANY>**

- Съдържанието му може да бъде произволно:
  - да бъде празен или не
  - да съдържа текст или не
  - да има смесено съдържание
  - или различни под-елементи в произволен ред
  - и т.н.

Моделни групи

- Дефинират съдържанието на елементите
  - **<!ELEMENT person (name, e-mail*)>**
- Задават йерархията на елемент
  - **<!ELEMENT name (fname, surname)>**
    <!ELEMENT fname (#PCDATA)>
    <!ELEMENT surname (#PCDATA)>
    <!ELEMENT e-mail (#PCDATA)>

- Управляват организацията на елементите
  - Sequence connector - **','** - **(A, B, C)** [then]
  - Choice connector - **'|'** - **(A | B | C)** [or]

Смесено съдържание

- <!ELEMENT descr (#PCDATA | a | b)*>

- *#PCDATA трябва да предхожда останалата част от дефиницията*

- Използваме ‘*choice*’ за разделяне на елементите

Гранулярността на XML структурата се отразява в DTD

- ```
  <PERSON>
     <NAME>Captain John Smith</NAME>
  </PERSON>
  ```

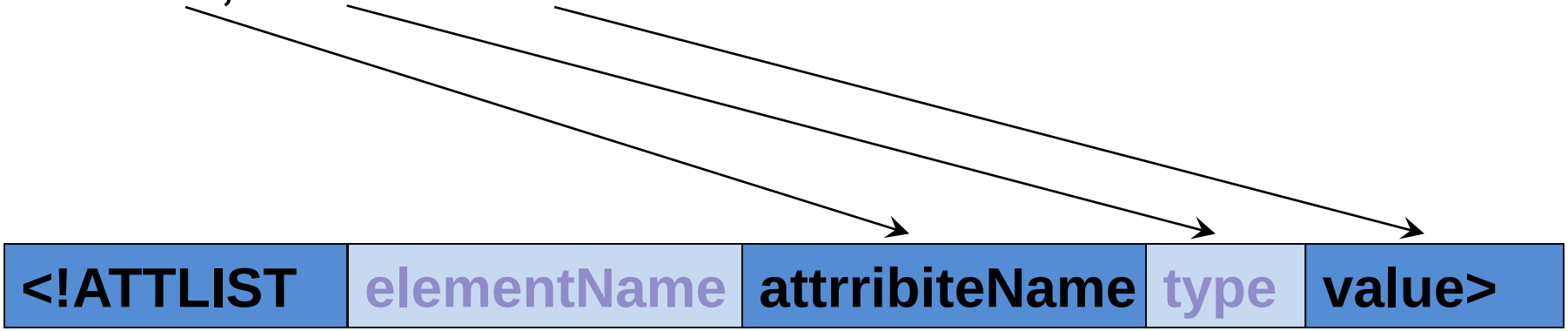
ИЛИ

- ```
  <PERSON>
    <TITLE>Captain</TITLE>
    <FORENAME>John</FORENAME>
    <SURNAME>Smith</SURNAME>
  </PERSON>
  ```

- Въпрос: *как да изберем гранулярността на нашия DTD документ*?

Деклариране на атрибути в DTD

- Декларацията на атрибут:
- ключова дума **ATTLIST**
- име на елемент, към който е свързан атрибута;
- списък на декларирани атрибути
- За всеки атрибут задължително се задава име, тип и стойност.



Още за атрибутите

- Декларация на множество атрибути:
- **<!ATTLIST myElementName**
- **attrName1 TYPE VALUE**
- **attrName2 TYPE VALUE**
- **attrName3 TYPE VALUE>**
- **<!ATTLIST myElementName**
- **attrName4 TYPE VALUE>**
- Дефинираните пространствата от имена, напр. **xmlns:tasks ="http://aemon.net/project/tasks"**, също се третират като атрибути.
- Атрибутите без префикс на пространство от имена не се отнасят към пространството от имена по подразбиране.

Честота на срещане на атрибути?

- За елементите използваме следните *оператори* за честота на срещане:
  - **?** – със значение нула или веднъж,
  - ***** - със значение нула или повече пъти,
  - **+** - веднъж или повече пъти.
- Тези *оператори за честота* се поставят след съответния елемент / елементи така:
  - **<!ELEMENT element-name**
    (child_1*, ((child_2, child_3?) | child_4+)*,...)>
- Въпрос: *може ли да ги ползваме за атрибути?*

Име и тип на атрибут

- Име на атрибут
  - `<!ATTLIST tag name type default>`
  - `<!ATTLIST tag first_attr …`
    `secon_attr …`
    `third_attr … >`
- Тип на атрибут

`CDATA`	`ID`
`NMTOKEN`	`IDREF`
`NMTOKENS`	`IDREFS`
`ENTITY`	`NOTATION`
`ENTITIES`	`name group`

Повече за типовете

- CDATA (non-parsed)
  - Character data
- NMTOKEN
  - Single token
- NMTOKENS
  - Multiple tokens
- ENTITY
  - Attribute is entity ref
- ENTITIES
  - Multiple entity ref's
- ID
  - Unique ID
- IDREF
  - Match to given unique ID defined elsewhere in the XML.
- IDREFS
  - Match to multiple ID's
- NOTATION
  - Describe non-XML data
- Name group
  - (enumeration) – restricted list

ID and IDREF(S) have to start with a letter or ':' or '_'. The rest of the characters must be numbers, letters, ':', '_', '-', or '.; cannot contain any spaces.

Примери за типове на атрибути

- CDATA
  - `name = "<Tom Jones>"`
- NMTOKEN
  - `color="red"`
- NMTOKENS
  - `values="12 15 34"`
- ENTITY
  - `photo="MyPic"`
- ENTITIES
  - `photos="pic1 pic2"`
- ID
  - `ID = "P09567"`
- IDREF
  - `IDREF="P09567"`
- IDREFS
  - `IDREFS="A01 A02"`
- NOTATION
  - `FORMAT="TeX"`
- Name group
  - `coord="X"`

Дефиниции на типове на атрибути

- CDATA
  - `<!ATTLIST person name CDATA … >`
- NMTOKEN (XML NameChar symbols only)
  - `<!ATTLIST mug color NMTOKEN … >`
- NMTOKENS
  - `<!ATTLIST temp val's NMTOKENS … >`
- ENTITY
  - `<!ATTLIST person photo ENTITY … >`
- ENTITIES
  - `<!ATTLIST album photos ENTITIES …>`
- ID
  - `<!ATTLIST person id ID … >`
- IDREF
  - `<!ATTLIST person father IDREF … >`
- IDREFS
  - `<!ATTLIST person children IDREFS … >`
- NOTATION
  - `<!ATTLIST image format NOTATION (TeX|TIFF) …>`
- Name group (enum)
  - `<!ATTLIST point title_point (Mr|Ms|Mrs|Rev|Dr) … >`

Указания за стойност на атрибут

- **#REQUIRED** Трябва да бъде зададен
- **#IMPLIED** Може да бъде зададен
- "***default***" Стойност по подразбиране, ако не е зададен
- **#FIXED** Само една разрешена стойност

Синтаксис:

```
<!ATTLIST  elementTag   attrName   type   value>
```

Пример:

```
<!ATTLIST seqlist sepchar NMTOKEN      #REQUIRED
                  type    (alpha | num) "num"
                  ver     CDATA    #FIXED "1.0"
>
```

Атрибут от тип ID

- Стойността на атрибут от тип ID трябва да е уникална за всички ID атрибути, декларирани в XML документа.
- Елемент може да има само един атрибут от тип ID, декларирана за него.
- Атрибут, деклариран от тип ID, трябва да бъде дефиниран като **#IMPLIED** или **#REQUIRED**.
- Стойността на атрибут от тип IDREF трябва да съответства на стойност на атрибут ID, дефинирана другаде в XML документа.

Атрибути от тип ID и IDREF

…

<!ELEMENT artist EMPTY>

<!ATTLIST artist name CDATA #REQUIRED>

<!ATTLIST artist **artistID ID #REQUIRED**>

<!ELEMENT album EMPTY>

<!ATTLIST album name CDATA #REQUIRED>

<!ATTLIST album **albumArtistID IDREF #IMPLIED**>

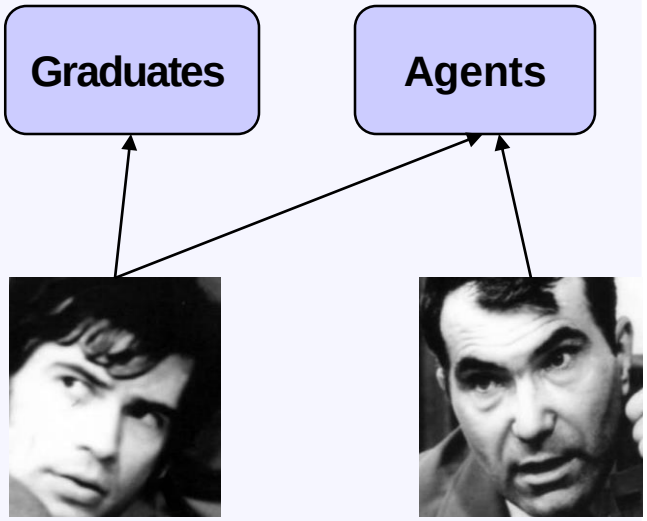
…

<artist name="Nick Cave" artistID="**NC**"/>

<album name="Murder Ballads" albumArtistID="**NC**"/>

…

```
<?xml version="1.0"?>
<!DOCTYPE characters [
  <!ELEMENT characters (character*, groups)>
  <!ELEMENT character (name, description?)>
  <!ELEMENT name EMPTY>
  <!ELEMENT descr (#PCDATA)>
  <!ELEMENT groups (group*)>
  <!ELEMENT group (#PCDATA)>
  <!ATTLIST name groups IDREFS #REQUIRED>
  <!ATTLIST name first CDATA #IMPLIED>
  <!ATTLIST name last CDATA #REQUIRED>
  <!ATTLIST group groupID ID #REQUIRED>
  <!ATTLIST group status (important|normal) "important">
]>
<characters>
  <character>
    <name last="Deyanov“ groups="MNI5567 MNI7890"/>
    <descr>Played by Stefan Danaylov<descr/>
</character>
  <character>
    <name first="Mitko" last="Bombata“ groups="MNI7890"/>
    <descr>Played by Grigor Vachkov<descr/>
</character>
  <groups>
    <group groupID="MNI5567" status="important">Graduates</group>
    <group groupID="MNI7890" status="normal">Agents</group>
  </groups>
</characters>
```



Псевдо-атрибути

- Имат вида ***xml:pseudoAttrName***
- Стойността на псевдо-атрибутите трябва да бъде предефинирана, като напр. тази на **xml:lang** или **xml:space**, които съответно се използват за дефиниране на код на език на съдържанието на елемента съгласно ISO-639 и за запазване (`xml:space="preserve"`) или не на допълнителни интервали в съдържанието на елемента.
- Друг псевдо-атрибут е **xml:id**, който задава условие за уникална стойност, без това да е необходимо да се декларира в DTD.
- **xml:base** предефинира стойността на **xlink:href** като относителен URI, като го допълва отпред с базовия адрес.

Пример

```
<poem xml:space="default">
  <author>
      <givenName> Димчо </givenName>
      <familyName>Дебелянов</familyName>
  </author>
  <verse xml:space="preserve">
      <line>   И   ето   скръб </line>
      <line>крила над мен привежда</line>
      <signature xml:space="default">
            Д. Дебелянов</signature>
  </verse>
</poem>
```

Интервали в атрибутите

- Въпреки че XML процесорите могат запазват всички интервали (бели полета) в **съдържанието на елемента**, те често го нормализират в **стойностите на атрибутите**.

- Табулации, пренос на нов ред, както и празни пространства се отчитат като един интервал.

- Ако документът е с дефиниран DTD, това подрязване ще се извърши за всички атрибути, които не са от тип CDATA.

xml:base - пример

```
<?xml version="1.0"?>
<doc xml:base="http://example.org/today/"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <paragraph>See <link xlink:type="simple" xlink:href="new.xml">
                        What's new</link>!
    </paragraph>
    <paragraph>Check out the hot picks of the day!</paragraph>
    …
  </body>
</doc>
```

В този случай връзката "What's new" ще сочи към http://example.org/today/new.xml

Декларация на коментари

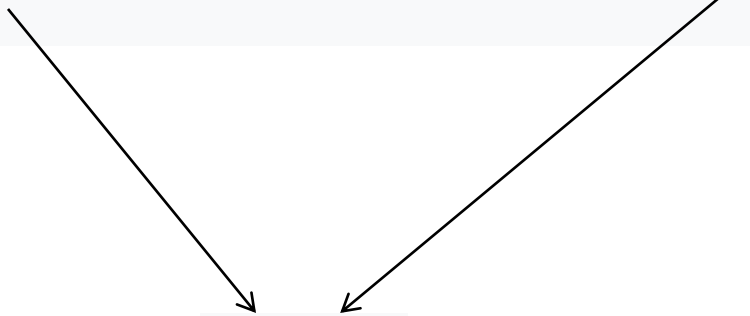
- Коментарите не са част от съдържанието на документа
  - `<!-- A comment -->`
- Забранено е ползването на '--' в коментар
- Коментар не може да включва други декларации на коментари в себе си

Декларация на символна секция

- За документа:
  - **`Press <<<ENTER>>>`**
- Декларираме:
  - **`<![CDATA[Press <<<ENTER>>>]]>`**
- CDATA секциите не могат да се влагат една в друга
- Дискусия: как да представим в **CDATA** стринга **"]]>"** ?

Символна секция със стринга "]]>"

```
<![CDATA[]]]> <![CDATA[>]]>
```



Използване на символни секции при inline <script> и <style> елементи в XHTML документи

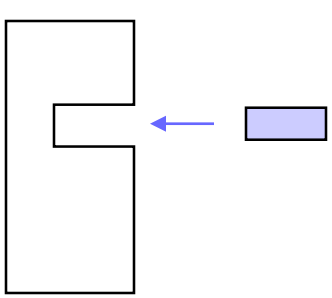
```
<script type="text/javascript">
 //<![CDATA[
 document.write("<");
 //]]>
</script>
<style type="text/css">
 /*<![CDATA[*/
 body { background-image: url(“111.png?width=300&height=300") }
 /*]]>*/
</style>
```

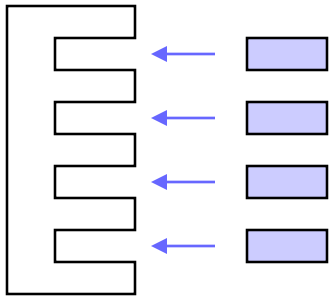
Дискусия

- Каква е разликата при интерпретирането на двата реда по-долу:
- **<sender>**Емил Боев**</sender>**
- и
- **<sender>**Емил Боев**</sender>**

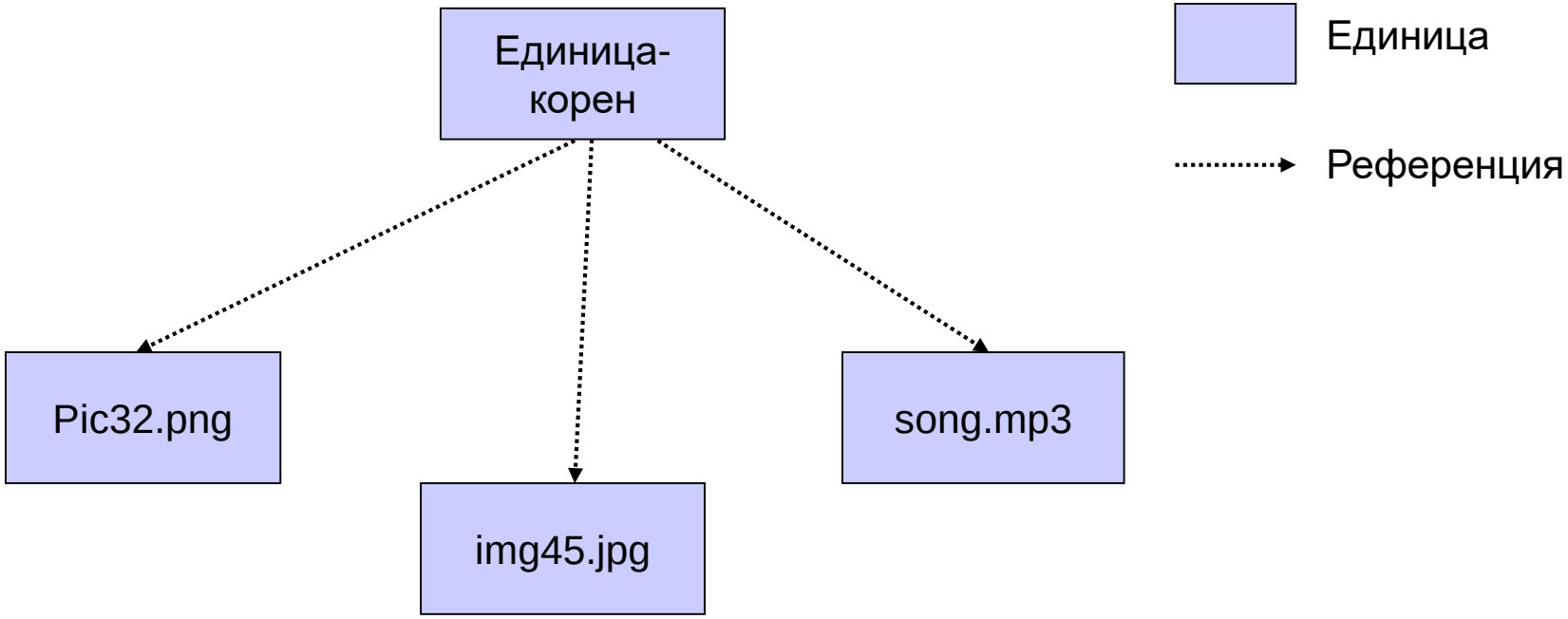
Единици (Entities)

- XML документът може да бъде разпределен сред голям брой файлове. Всяка част от организирането на информацията се нарича единица (entity).
- Единицата има име, да я идентифицира. Тя се дефинира чрез специална ***декларация*** (**ENTITY**) и се използва през ***референция***, която я указва.
- В един XML документ, **ENTITY** референциите се използват като указател към текст или външен ресурс.





Единици (XML Entities)



Кога да използваме единици

- Единиците подобряват четимостта на XML документите:
  - Повторението на един и същи текст на много места се заменя с кратък запис
  - Могат да имат различно представяне
  - Разделянето на документа го прави по-лесно управляем
  - Може да се използват и други, не-XML формати

Типове единици

- **Общи** (General)
  - *Декларирани в DTD; Реферирани в* XML документите
- **Параметрични**
  - *Декларирани в DTD;*
  - Реферирани *единствено* в декларациите на DTD;
- **Вътрешни** (Internal)
  - Съхраняват се в XML документа
  - Текстово съдържание
- **Външни** (External)
  - Съхраняват се извън XML документа
  - Текстово или двоично съдържание
  - Могат да групират други единици

Общи единици (General Entities)

- Декларирани в 'Document Type Declaration'
  - <!DOCTYPE My_XML_Doc [
    <!ENTITY name "*replacement*">
    ]>
- Използват се в XML - пример:
  - В DTD: <!ENTITY xml "eXtensible Markup Language">
  - В XML: The &xml; includes entities
  - Резултат: The eXtensible Markup Language includes entities

Предефинирани *общи единици*

Могат да бъдат използвани без да е необходимо да се специфицират. Те са:

- **'** - представя знака *апостроф* (') – резервиран заедно с (") за ограждане на стойности на атрибути;
- **"** - представя знака *кавички* (") - резервиран заедно с (') за ограждане на стойности на атрибути;
- **&** - представя знака *амперсанд* (&)- резервиран за задаване на единици;
- **>** - представя знака *по-голямо* (>) - резервиран заедно с (<) за ограждане на елементи;
- **<** - представя знака *по-малко* (<) - резервиран заедно с (>) за ограждане на елементи.

Параметрични (Parameter Entities)

- Декларирани в 'Document Type Declaration'
  - <!DOCTYPE My_XML_Doc [
    <!ENTITY % name "*replacement*">
    ]>
- **Използват се в DTD:**
  - <!ENTITY % param "(para | list)">
  - <!ELEMENT section (%param;)*>
  - вместо
    - <!ELEMENT section (para | list)*>

Вътрешни единици (Internal Entities)

- Реферират текст, който е дефиниран в DTD документа. Дефиницията им започва с ключовата дума **ENTITY** следва името на единицата и накрая нейната стойност заградена в кавички.
- **<?xml version="1.0" standalone="yes" ?>**
- **<!DOCTYPE name [**
- **<!ELEMENT name (#PCDATA)>**
- **<!ENTITY myFirstName "Elen">**
- **<!ENTITY myLastName "Lenon">**
- **]>**
- **<name>&myFirstName; &myLastName;</name>**

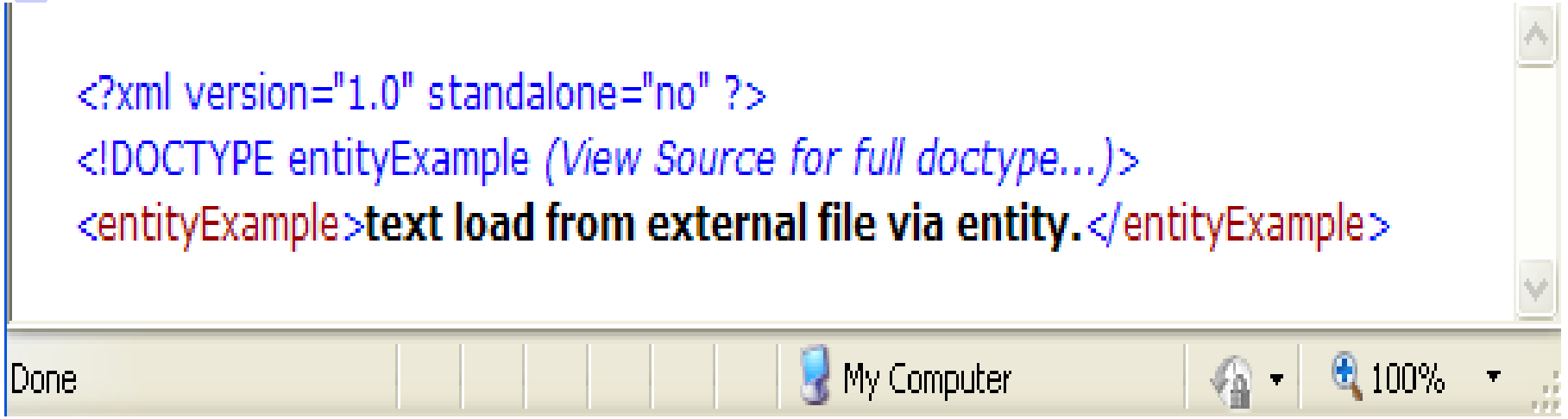
Външни (External Entities)

Съдържат XML съдържание – реферират към данни, които един XML процесор трябва да може да обработи

- А. ***частни*** – при дефиниция се използва ключовата дума *SYSTEM.* Те са предназначени и достъпни за определена група от хора:
- **<!ENTITY entity_name SYSTEM "URI">**
- Б**. *публични*** - използват ключовата дума *PUBLIC*. Те са налични в Интернет и могат да бъдат използвани от всеки:
- **<!ENTITY entity_name PUBLIC "PUBLIC_ID" "URI">**

Външна частна единица

- **<?xml version="1.0" standalone="no" ?>**
- **<!DOCTYPE entityExample [**
- **<!ELEMENT entityExample (#PCDATA)>**
- **<!ENTITY entityData SYSTEM "example.txt">**
- **]>**
- **<entityExample>&entityData;</entityExample>**



Външна публична единица

- **<!ENTITY entity_name PUBLIC "PUBLIC_ID" "URI">**
- **entity_name** е името, с което съответната единица ще бъде използвана в XML документа, **PUBLIC_ID** може да се използва от XML процесора, за да генерира алтернативен URL адрес и ако той не може да бъде намерен, то се използва дефинирания адрес в **URI**.
- **<?xml version="1.0" standalone="no" ?>**
- **<!DOCTYPE entityExample [**
- **<!ELEMENT entityExample (#PCDATA)>**
- **<!ENTITY entityData PUBLIC "-//W3C//TEXT entity//BG" "http://www.w3.org/xmlspec/entity.xml" >**
- **]>**
- **<entityExample>&entityData;</entityExample>**

Външни единици, съдържащи не-XML съдържание

- За вграждане на не-XML данни (напр. графика), използваме външно ***unparsed entity***
- Реферират към данни, които XML процесора не трябва да обработва
- Могат да бъдат частни и публични
- След ключовата дума *NDATA* следвана от име на задължително дефинирана в DTD документа нотация:

- **<!ENTITY entityName SYSTEM "URI" NDATA notation_name>**
- **<!ENTITY entityName PUBLIC "PUBLIC_ID" "URI" NDATA notation_name>**

Външни единици, съдържащи не-XML съдържание - пример

```
<?xml version="1.0"?>
<!DOCTYPE company [
  <!ELEMENT company (logo) >
  <!ELEMENT logo EMPTY>
  <!ATTLIST logo url ENTITY #REQUIRED>
  <!ENTITY company_logo SYSTEM " logo.gif" NDATA GIF>
  <!NOTATION  GIF PUBLIC “image/gif ">
]>
<company>
  <logo url="company_logo"/>
</company>
```

Нотация (Notation)

- Нотацията се използва, за да специфицира различни от XML данни като например файлове от тип image/gif, image/jpeg и др.
- Специфичното при този тип атрибут е, че при декларация трябва да бъдат изброени стойностите, които атрибутът може да приема и всяка една от тях трябва да бъде име на декларирана в DTD документа нотация.

Декларация на нотация (Notation)

- Описва external non-XML entity
- Използва се с **NDATA**

```
<!NOTATION jpeg SYSTEM "image/jpeg">
<!ENTITY turing_getting_off_bus
          SYSTEM "http://www.tur.org.uk/aa.jpg"
          NDATA jpeg>
<!ELEMENT image EMPTY>
<!ATTLIST image source ENTITY #REQUIRED>

<image source="turing_getting_off_bus"/>
```

- Друг начин за това е да се ползва XLink, подобно на задаването в HTML

Ограничения при единиците 1/2

- За общите текстови единици
  - Не могат да се използват рекурсивно
  - Могат да се ползват в елементно съдържание
    - **<para> … &ent; … </para>**
  - Допустими са в атрибутно съдържание
    - **<para name="&ent;"> … </para>**
  - Домустими са в други вътрешни единици
    - **<!ENTITY cod "&ent;">**
  - Не могат да се ползват в други части на DTD

Ограничения при единиците 2/2

- Двоично съдържание
  - Ако съдържанието не е XML, такава единица не може да се ползва като референция
    - Error - **<!ELEMENT sec (para|&photo;)>**
    - Error - **<para> &photo; </para>**
  - Двоичните единици *могат да бъдат **само** атрибут* от тип ENTITY
    - **<!ENTITY photo SYSTEM "photo.tif" NDATA TIFF>**

```
...
<!ELEMENT pic (#PCDATA)>
<!ATTLIST pic name ENTITY #REQUIRED>
```

<pic name="photo"/>

Условни секции 1/3

- Единиците от параметричен тип могат успешно да бъдат използвани и при т. нар. *условно* DTD.
- То дава възможност в един DTD документ да има *условни* части, една от които е означена с ключовата дума **INCLUDE**, а останалите с **IGNORE**.
- В този случай от всички *условни* части на DTD за XML документа ще бъде валидна тази означена с INCLUDE.
  - **<![INCLUDE[…]]>**
  - **<![IGNORE[…]]>**

Условни секции 2/3

- Включване на декларации:

```
<!ENTITY % variant "INCLUDE">

<![%variant;[
  <!ENTITY % Text "(#PCDATA|temp)">
]]>
```

- Изключване на декларации:

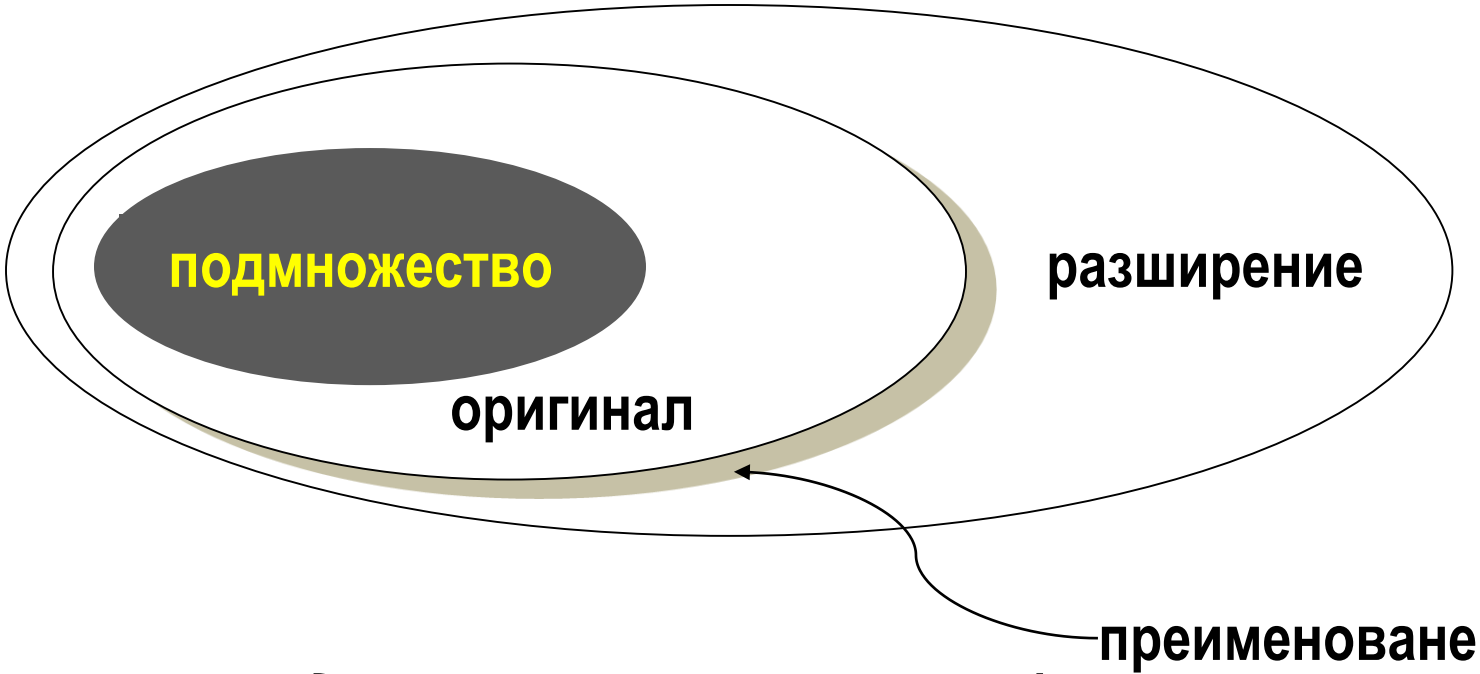
```
<!ENTITY % variant "IGNORE">

<![%variant;[
  <!ENTITY % Text "(#PCDATA|temp)">
]]>
```

Условни секции 3/3

- <!ENTITY % big.DTD "IGNORE">
- <!ENTITY % small.DTD "INCLUDE">
- ⋮
- <![%big.DTD; [
- <!ENTITY % blocks "para|excerpt|epigraph">
-]]>
- <![%small.DTD; [
- <!ENTITY % blocks "para|excerpt">
-]]>
- ⋮
- <![%big.DTD; [
- <!ELEMENT epigraph (#PCDATA)>
-]]>

Проектиране на DTD с цел многократно използване



Оригинална дефиниция на атрибут:

```
<!ATTLIST document status CDATA #IMPLIED >
```

Подмножество:

```
<!ATTLIST document status (draft|final) #IMPLIED>
```

Техники за управление на персонализация на дефинициите 1/2

- Работа с модулни DTD. Принципите (критериите) на Meyer за изграждане на модулни системи са приложими и тук
- Персонализируеми модели на съдържанието - чрез начини за поставяне на контейнери в елементните декларации и в списъците атрибути
- Условно маркиране декларации - как да се използват маркираните секции с цел условно включване на декларации за маркиране

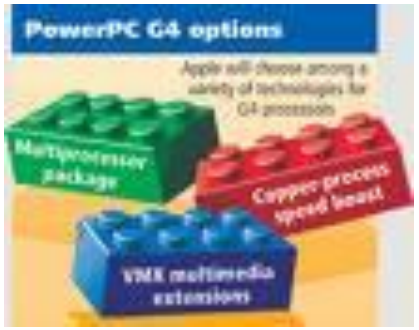
Техники за управление на персонализация на дефинициите 2/2

- Работа с персонализируеми имена за маркиране - чрез техники за персонализиране на имената на елементите и други имена, напр:

```
<!ENTITY % title "titulo">
<!ELEMENT %title (#PCDATA)>
<!ATTLIST %title id ID #IMPLIED >
...
<!ELEMENT div ((%title), para+, subdiv*)>
```

Критерии на Meyer за оценка на системната модулност (модулни DTD)

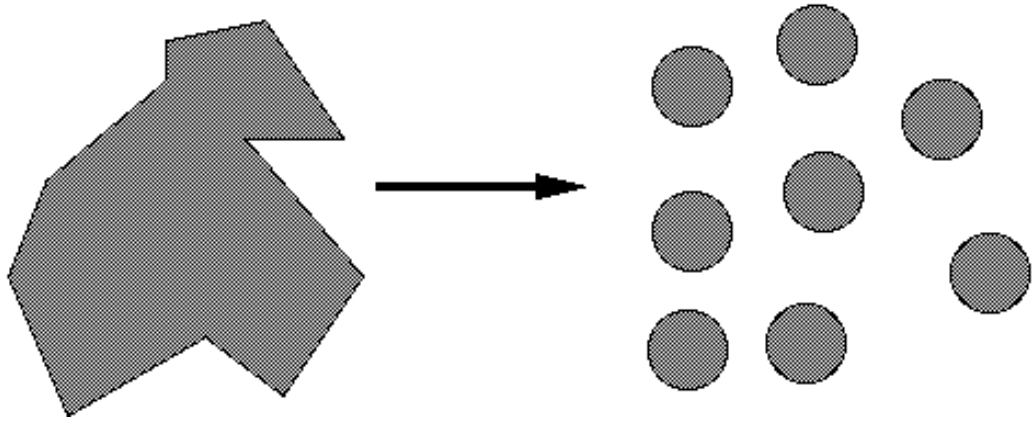
- Декомпозируемост
- Композируемост
- Разбираемост
- Непрекъснатост (континюитет)
- Протекция



Декомпозируемост (Decomposability)

- Идея: проблемът да се разложи на по-малки под-проблеми, които могат да бъдат решени отделно

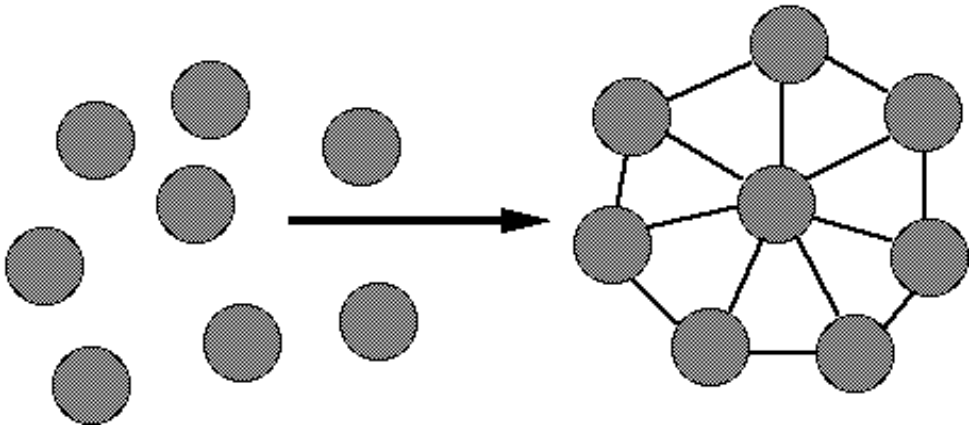
- Пример: Top-Down дизайн
- Контрапример: Initialization модул



Композируемост

- Идея: да се комбинират свободно модули за създаване на нови системи

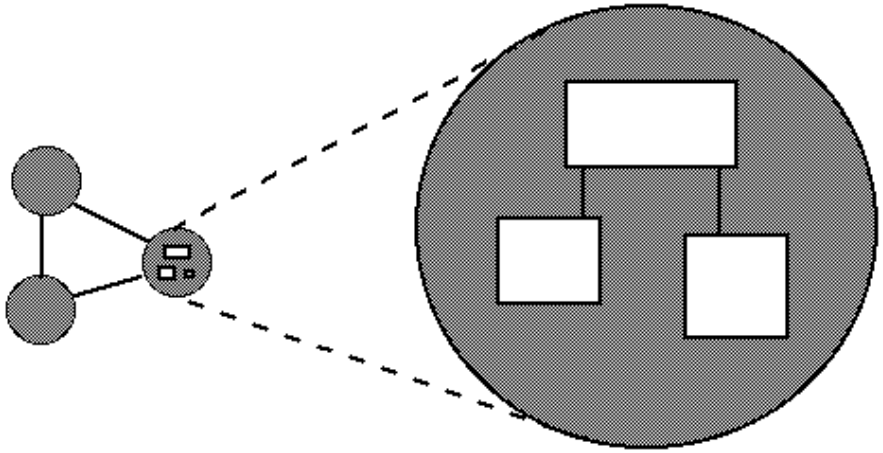
- Пример: Math libraries, Unix command & pipes
- Контрапример: ?



Разбираемост

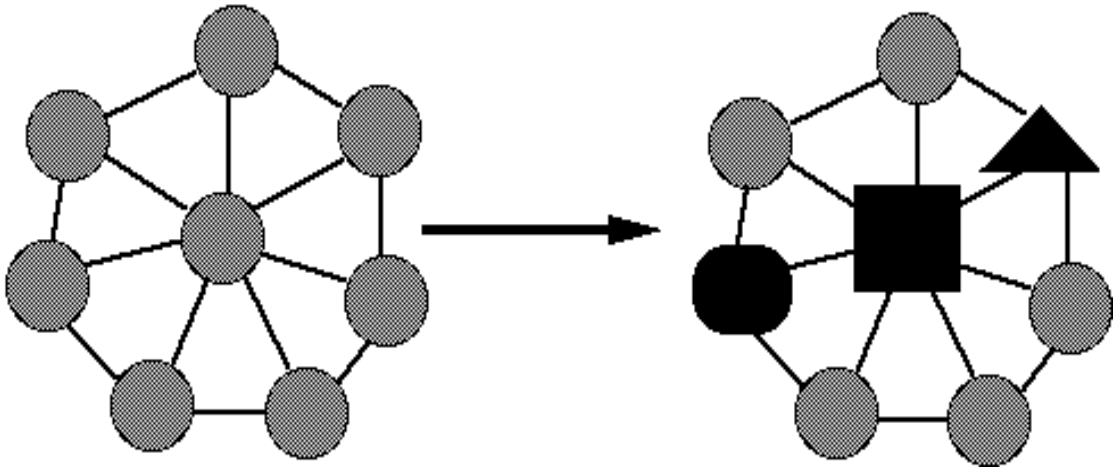
- Идея: отделните модули да бъдат разбираеми за читателя

- Пример: ?
- Контрапример: Sequential Dependencies



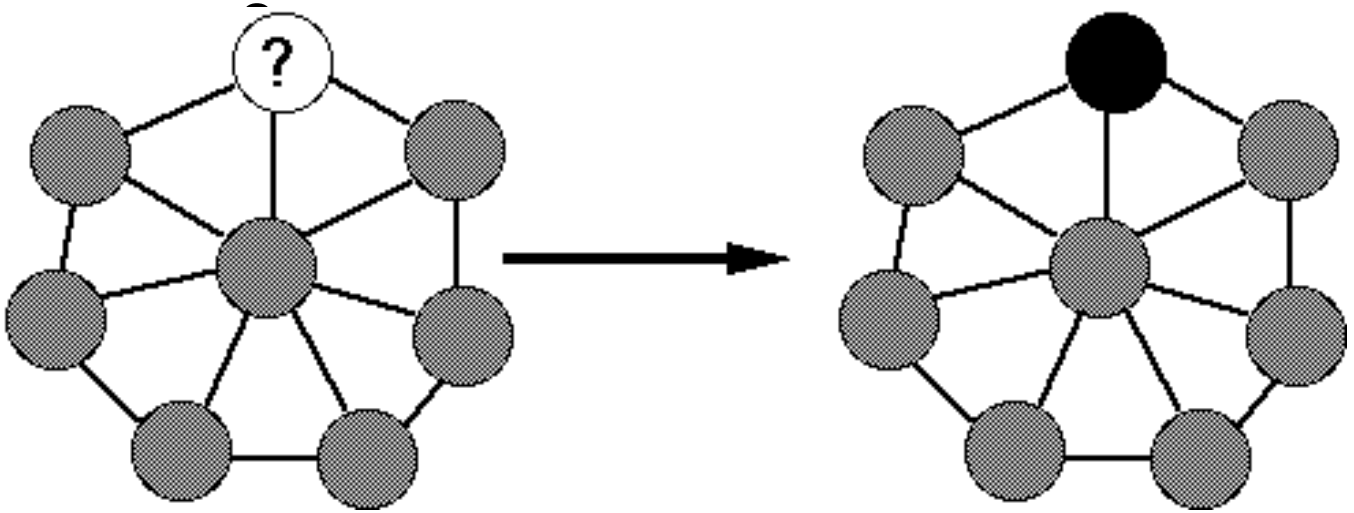
Непрекъснатост (континюитет)

- Идея: малката промяна в спецификацията да има за резултат:
  - - промени в само няколко модула
  - - да не засяга архитектурата
- Пример: константи -> const MaxSize = 100
- Контрапример: ?



Протекция

- Идея: влиянието на ненормално състояние по време на изпълнение се ограничава до няколко модули
- Пример: Validating input
- Контраг



Два важни принципа за разработка на софтуер

KISS: Keep it *simple & stupid*

Поддържа:

- Разбираемост
- Композируемост
- Декомпозируемост

Small **is Beautiful**

Def.: Upper bound for average size (lines of code) of an operation [Lorenz'93]:

Smalltalk – 8

C++ - 24

Поддържа:

- Декомпозируемост
- Композируемост
- Разбираемост

Web based e-Learning (free!)

- **W3 School: DTD School**
  - **http://www.w3schools.com/dtd/**
- **Introduction to DTD**
  - **An introduction to the XML DTD, and why you should use it.**
- **DTD - XML Building Blocks**
  - **The XML building blocks that can be defined in a DTD.**
- **DTD Elements**
  - **How to define the legal elements of an XML document using DTD.**
- **DTD Attributes**
  - **How to define the legal attributes of XML elements using DTD.**
- **DTD Entities**
  - **How to define XML entities using DTD.**
- **DTD Validation**
  - **How to test for DTD errors while loading XML documents.**
- **DTD Examples**
  - **Some real world DTD examples.**


?
Q
&
&
A
!