

Șabloane de proiectare XML

Sabin-Corneliu Buraga
Facultatea de Informatică

busaco@infoiasi.ro
<http://www.infoiasi.ro/~busaco/>

cuprins

- Punerea problemei
- Ce este un *pattern*?
- *Pattern*-uri de proiectare a structurii XML
- *Pattern*-uri de proiectare a aplicațiilor utilizând XML



punerea problemei

- O problemă poate apărea frecvent
- Cei experimentați au găsit diverse soluții pentru problema în cauză, reușind să recunoască problema și să aleagă soluția (optimă) care poate fi aplicată într-un anumit context
- Apare astfel un șablon (*pattern*) de proiectare (soluționare) a problemelor

ce este un *pattern*?

- Un *pattern* \equiv o regulă care exprimă o relație dintre un **context**, o **problemă** și o **soluție** (Christopher Alexander, 1979)
 - Inițial, folosit în arhitectură
 - Ulterior, în proiectarea software (*object-oriented software design*):
 - pattern* \equiv “mind sized” chunk of information
 - W. Cunningham & K. Beck, 1987
 - “*Design Patterns*” (E. Gamma et al., 1995)



ce este un *pattern*?

- Un *pattern* poate descrie cunoștințele unui expert (pe baza experienței sale personale) în domeniul problemei în ceea ce privește recunoașterea problemei, a contextului și a soluției la acea problemă
- Un *pattern* nu reprezintă o regula fermă (uneori nu trebuie aplicat!)
- Este necesară adoptarea unui vocabular comun corespunzător domeniului problemei \Rightarrow *pattern language*

ce este un *pattern*?

- Tipuri de *pattern*-uri
 - De proiectare
 - De arhitectură
 - De analiză
 - De creație
 - De structură
 - De comportament



ce este un *pattern*?

- “Recunoașterea” unui *pattern* poate avea loc la unul dintre nivelurile:
 - De prezentare
 - De procesare (*business logic*)
 - De integrare



ce este un *pattern*?

- Specificarea unui *pattern*:
 - Numele
 - Rezumatul
 - Problema
 - Contextul
 - Soluția
 - Exemplele
 - Utilizările

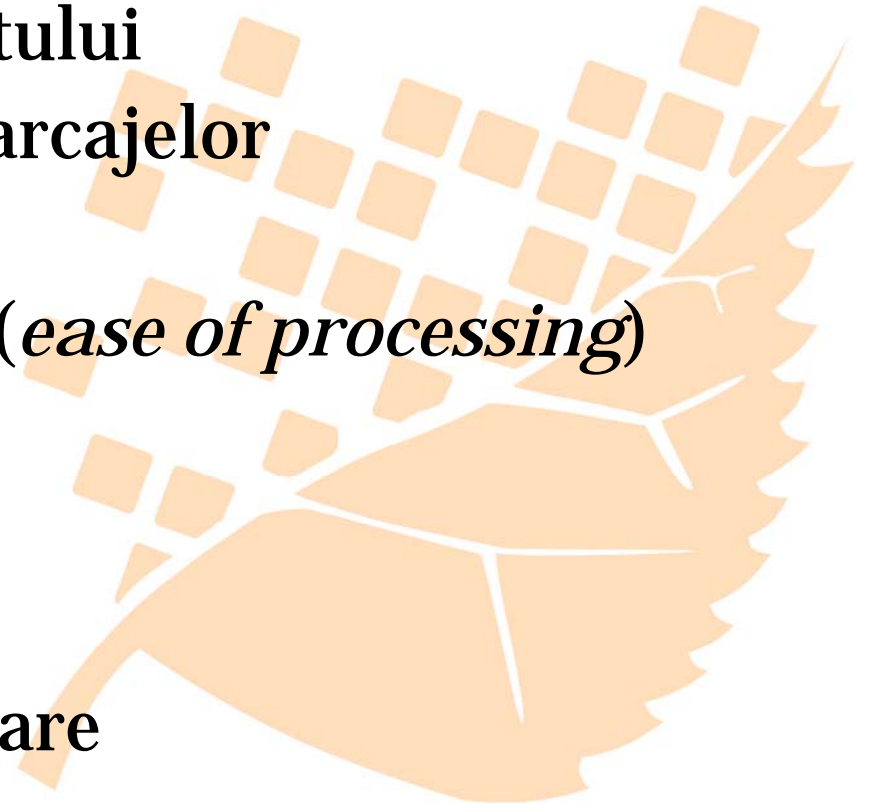


ce este un *pattern*?

- În cazul nostru, ne interesează meta-limbajul (familia) XML
- Întrebări:
 - Cum trebuie structurat un document XML pentru a stoca anumite date sau metadata?
 - Cum trebuie proiectată o aplicație care utilizează XML ca limbaj de specificare, stocare și/sau prezentare a datelor?

Pattern-uri XML structurale

- Proiectarea structurii unui document XML
 - Lungimea documentului
 - Ușurința folosirii marcajelor (*ease of authoring*)
 - Ușurința procesării (*ease of processing*)
 - Ușurința validării
 - Flexibilitatea
 - Consistența
 - Gradul de abstractizare



Pattern-uri XML structurale

- *XML Design Patterns*
 - Oportunitatea folosirii XML-ului: **Use XML**
 - Reutilizarea tipurilor de documente existente: **Reuse Document Type**
 - Alegerea elementului/elementelor rădăcină: **Multiple Document Types, Multi Root Document Types**
 - Stabilirea gradului de abstractizare: **Envelope, Short Understandable Names, Domain Element, Container Element, Collection Element,...**
 - Asocierea de metadate: **Separate Metadata & Data, Metadata in Separate Document, Head-Body, Metadata First**

Pattern-uri XML structurale

- *XML Design Patterns* – continuare
 - Organizarea documentului
 - Referențierea unor construcții:
Declare Before First Use
 - Aceeași informație referențiată în locuri multiple: **Flyweight**
 - Arbore (ierarhie) *versus* graf: **Marketplace**
 - Extinderea ulterioară: **Catch-All Element, Role Attribute, Extensible Content Model**
 - Asigurarea consistenței:
Common Attributes, Consistent Element Set

Pattern-uri XML structurale

- **Use XML**

- Determină situația în care XML e soluția viabilă de reprezentare a informației (semi)structurate
- Existența unor reprezentări multiple: binar, CSV, HTML, baze de date relaționale, obiecte serializate, XML
- XML poate fi o soluție adecvată când:
 - Conținutul (datele) trebuie separate de formatare
 - Datele trebuie partajate între aplicații, organizații,...
 - Reprezentarea să poată fi înțeleasă și de oameni
 - Reprezentarea să poată fi ușor procesată de calculator, independent de platformă și de limbaj

Pattern-uri XML structurale

- **Use XML**

- Factori care trebuie considerați: simplitatea, extensibilitatea, interoperabilitatea, existența instrumentelor de procesare, transformarea facilă în alte reprezentări, ușurința validării, existența standardizării
- XML ca limbaj pentru reprezentarea atât a datelor, cât și a metadatelor

```
<rdf:Description
```

```
  about="http://s.infoiasi.ro/studenti.cgi?matricol=33">
```

```
  <nume porecla="Sammy">Mihai Alexandru Serea</nume>
```

```
  <orar> <zi tip="luni">... </zi> </orar>
```

```
</rdf:Description>
```

Pattern-uri XML structurale

- **Short Understandable Names**
 - Numele elementelor & atributelor trebuie să fie scurte și ușor de înțeles atât de autori, cât și de dezvoltătorii soft-ului de procesat documentul
 - *Pattern*-ul poate fi utilizat pentru orice tip de document
 - Numele prea scurte sunt dificil de înțeles, dar reduc lungimea documentelor
 - Convenții de numire: `<nume_tag>`, `<NumeTag>`
 - Exemple:
 - `` `` vs. `` `<table>`
 - `<DescriereaEchipamentului Codul="">` vs. `<desec c="">`

Pattern-uri XML structurale

- **Content Element**
 - Un container grupează elemente (copil) înrudite
 - Problema: numeroase elemente care apar pe același nivel în document și care pot fi divizate în grupuri distincte
 - *Pattern*-ul ajuta la structurarea documentului, dar e foarte general (pot exista *pattern*-uri mai specializate, derivate din acesta)
 - *Pattern*-ul adaugă un nivel de abstractizare, gruparea elementelor oferind informații semantice suplimentare (e.g., asocierea de metadata unui grup de elemente)
 - *Pattern*-uri înrudite: **Head-Body**, **Collection Element**

Pattern-uri XML structurale

- **Content Element**

```

<config>
  <ram>256</ram>
  <hdd tip="...">80</hdd>
  <parser limbaj="C++">
    Xerces</parser>
</editor>vim</editor>
<parser limbaj="Perl">
  Expat</parser>
</config>
    
```

```

<config>
  <hardware>
    <ram>256</ram>
    <hdd tip="...">80</hdd>
  </hardware>
  <software>
    <parser limbaj="C++">
      Xerces</parser>
    <parser limbaj="Perl">
      Expat</parser>
    <editor>vim</editor>
  </software>
</config>
    
```

Pattern-uri XML structurale

- **Collection Element**

- Creează un element al cărui model de conținut permite doar instanțe de singur tip
- Problema: există un element care trebuie repetat la același nivel al documentului
- Context: gruparea pe categorii a elementelor, existența multor “frați” (*siblings*), asocierea de metadata etc.
- Soluție: crearea unui element conținând mai multe elemente de același tip
- Structura rezultată e mult mai ușor de procesat
- Dacă volumul de metadata este mare, se va putea utiliza *pattern*-ul **Head-Body**
- Exemple: **XHTML**, **RDF**, **DocBook**,...

Pattern-uri XML structurale

- **Envelope**

- Oferă un tip de document care va desemna un “plic” în care se vor putea stoca date XML arbitrare
- Problema: diferite seturi de date trebuie livrate unui sistem, într-o manieră consistentă
- Context: structura datelor din “plic” poate varia și nu e cunoscută la momentul creării sistemului
- *Pattern*-ul permite separarea diferitelor tipuri de conținuturi, oferind un mecanism de livrare a datelor XML; plicul nu interferează cu conținutul propriu-zis al mesajului transmis
- Exemplu: **SOAP** (Simple Object Access Protocol)

Pattern-uri XML structurale

- **Flyweight**

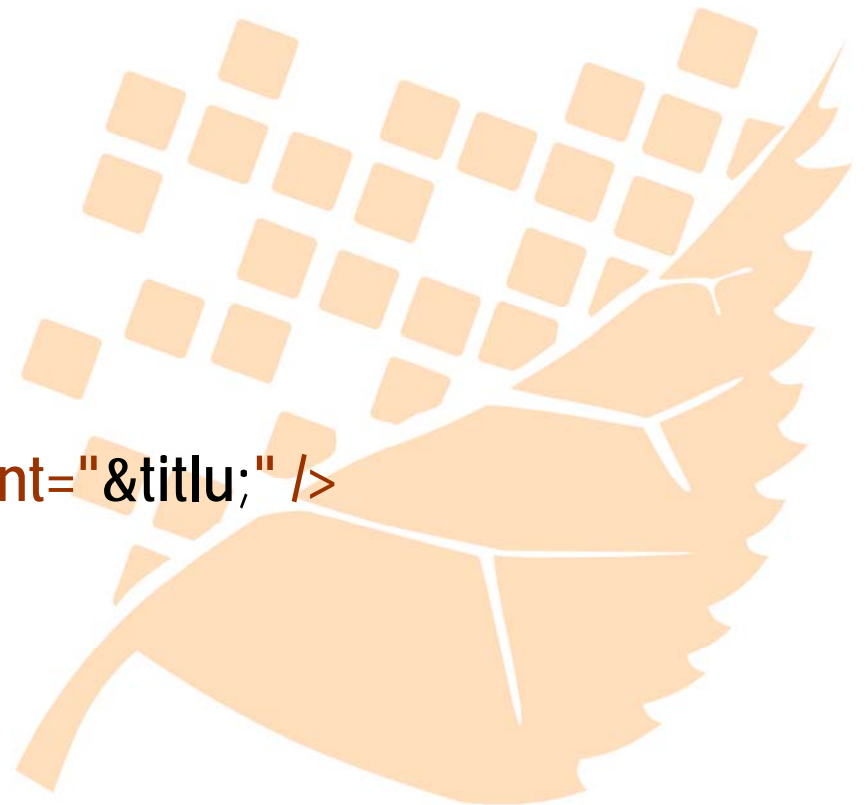
- Când aceeași informație este inclusă în diferite locuri în document, atunci ea poate fi plasată o singură dată și referită în locuri multiple
- Problema: plasarea repetată a acelorași date în locuri diferite poate cauza erori și dificultăți în mentenanța documentului; crește nejustificat lungimea documentului
- Soluție: utilizarea entităților XML (externe), folosirea datelor “*embed*” via XLink, utilizarea atributelor ID și IDREF etc.
- *Pattern*-ul mărește gradul de mentenanță și modularizare, dar poate afecta abilitatea de înțelegere a documentului

Pattern-uri XML structurale

- **Flyweight**

```

<?xml version="1.0" ?>
<!DOCTYPE Titlu [
<!ENTITY titlu "Situl WebGroup">
]>
<html>
<head>
  <title>&titlu;</title>
  <meta name="description" content="&titlu;" />
</head>
<body>
  <h1>&titlu; :: Salut!</h1>
</body>
</html>
  
```



Pattern-uri de proiectare XML

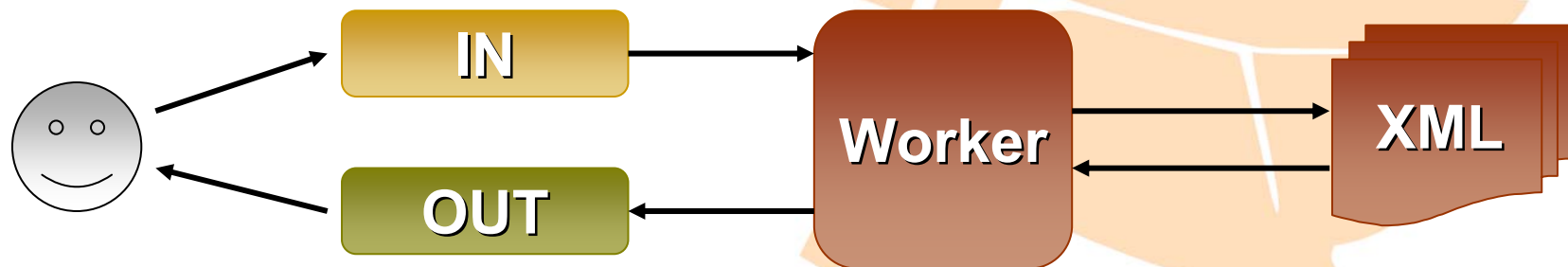
- Proiectarea unei aplicații Web (bazata pe tehnologiile XML)
 - Separarea dintre modul de **stocare** a datelor, **vizualizarea** lor și maniera de **procesare**
 - *Pattern*-ul consacrat:
Model-View-Controller (MVC)
- Problema:
disponibilitatea pe un sit Web a datelor provenite din surse XML distribuite

Pattern-uri de proiectare XML

- Exemple de *pattern*-uri de proiectare:
 - **XML In Out Tray** – rezolvă problema obținerii, procesării și returnării datelor XML, procesele interne de prelucrare fiind ascunse
 - **External Assistant** – procesul de generare a formatului de prezentare plecând de la XML este realizat de o aplicație externă
 - **Information Grouping** – soluționează problema grupării și prezentării datelor XML, indiferent de aplicația care generează aceste date
 - **XML Mediator** – rezolvă problema inter-operabilității dintre aplicații care utilizează documente XML cu structuri eterogene

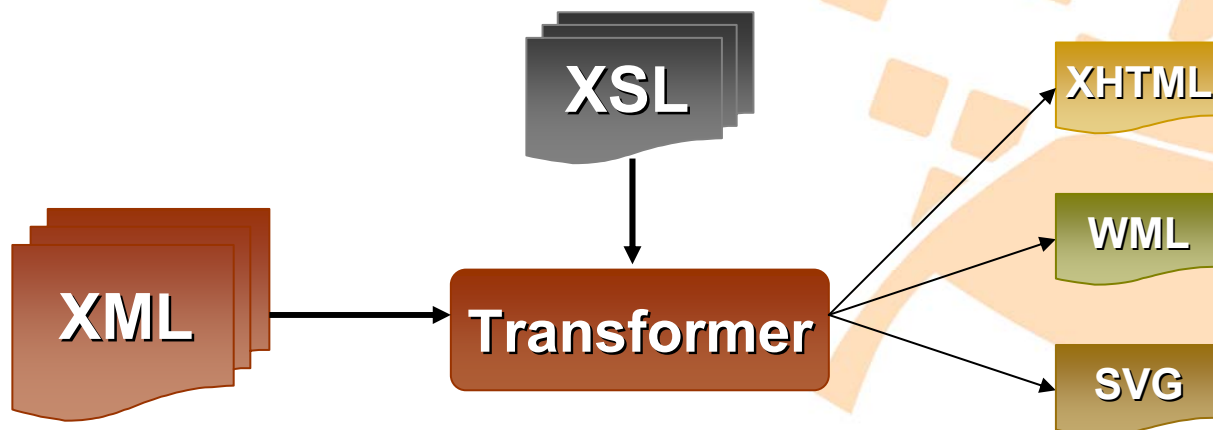
Pattern-uri de proiectare XML

- Exemple de *pattern*-uri de proiectare:
 - **XML In Out Tray**
 - Organizează activitatea componentelor implicate în procesele de colectare și de vizualizare a datelor
 - Scop: dezvoltarea de conexiuni între componente traversate de date XML păstrând o cuplare slabă și o coeziune ridicată (similar cu *pattern*-ul *Proxy*)
 - Existența unui “lipici” între componente
⇒ independența de limbaj/platformă



Pattern-uri de proiectare XML

- Exemple de *pattern*-uri de proiectare:
 - **Information Grouping**
 - Date la intrare documente XML, *pattern*-ul oferă o modalitate de a formata datele XML, grupate pe diverse criterii (asemănător lui *group by* din SQL)
 - Asigura separarea între formatul de reprezentare și cel de stocare a datelor, putând organiza informațiile într-un mod diferit de cel al stocării



Pattern-uri de proiectare XML

Exemple demonstrative
de aplicare a *pattern*-urilor XML



biblio


- E. Gamma *et al.*, *Design Patterns*, Addison-Wesley, 1995
- B. Daum, U. Merten, *System Architecture with XML*, Elsevier Science, 2003
- S. Buraga, *Semantic Web*, Matrix Rom, 2004
- S. Buraga (coord.), *Aplicații Web la cheie*, Polirom, 2003
- S. Buraga (coord.), *Situri Web la cheie*, Polirom, 2004

biblio

- www.w3.org/XML
- www.XMLpatterns.org
- www.mobileworkspace.com/xmlstds
- www.dublincore.org
- www.ZVON.org



rezumat

- Punerea problemei
 - Ce este un *pattern*?
 - *Pattern*-uri de proiectare a structurii XML
 - *Pattern*-uri de proiectare a aplicațiilor utilizând XML
- 

Mulțumiri pentru atenția acordată!

Întrebări?

