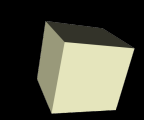


Effective **X**ML **C**ommunication **U**sing **S**chemas and **A**utomata

Pavol Rusnák

Motivation

Parse

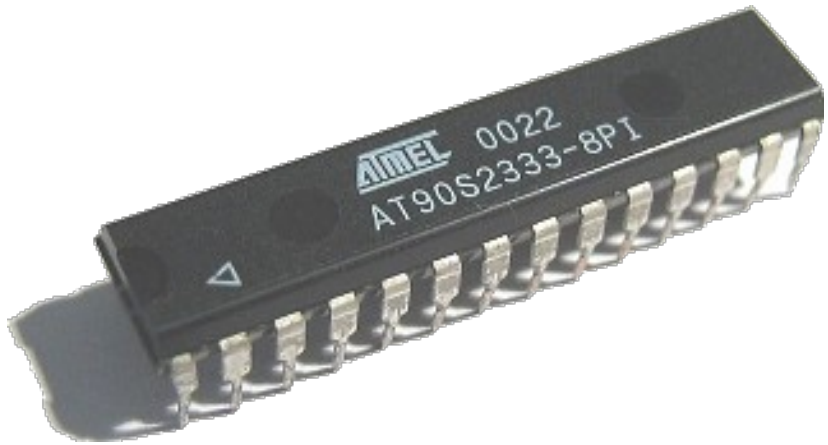
complex text-based messages

- **XML**
- **JSON**

on

Microcontrollers

- **slow CPU**
- **not enough RAM**



XML Message

```
<status>
  <fire>false</fire>
  <light>4</light>
  <temp>23.4</temp>
  <temp>18.3</temp>
  <temp>20.7</temp>
  <client>mitra</client>
  <client>spectra</client>
  ...
</status>
```

XML Schema

```
<element name="status">
  <complexType>
    <sequence>
      <element name="fire" type="s:boolean" />
      <element name="light" type="s:integer" />
      <element name="temp" type="s:float"
               minOccurs="3" maxOccurs="3" />
      <element name="client" type="s:string"
               minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
```

Process

XML Schema

→

Grammar

→

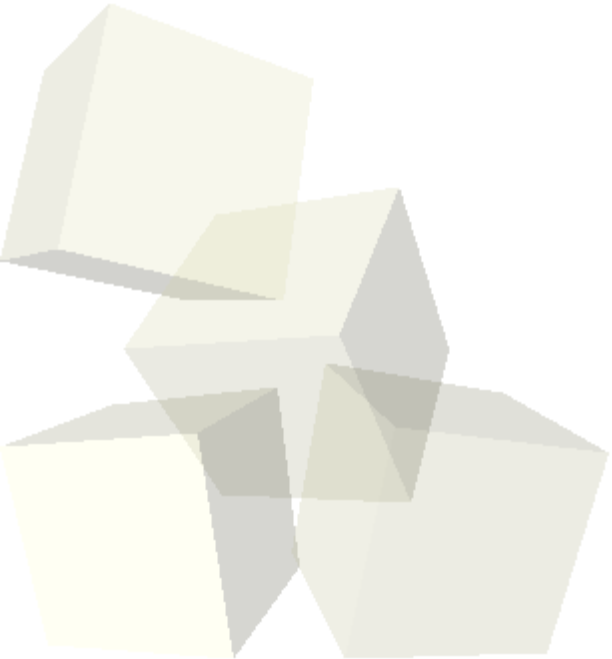
Finite State Machine

→

Code

→

Profit!



Element

```
<element name="tag" type="s:integer" />
```

A → <tag> ***B*** </tag>
B → *{integer}*

Optional Element

```
<element name="tag" type="s:integer"
         minOccurs="0" />
```

A → <tag> ***B*** </tag>
A → ε
B → *{integer}*

Repeating Element

```
<element name="tag" type="s:integer"
         minOccurs="1" maxOccurs="3" />
```

$$A \rightarrow BBB$$
$$A \rightarrow BB$$
$$A \rightarrow B$$
$$B \rightarrow \langle tag \rangle\ C\ \langle /tag \rangle$$
$$C \rightarrow \{integer\}$$

Infinitely Repeating Element

```
<element name="tag" type="s:integer"
 minOccurs="0" maxOccurs="unbounded"/>
```

$$A \rightarrow BA$$
$$A \rightarrow \varepsilon$$
$$B \rightarrow \langle tag \rangle\ C\ \langle /tag \rangle$$
$$C \rightarrow \{integer\}$$

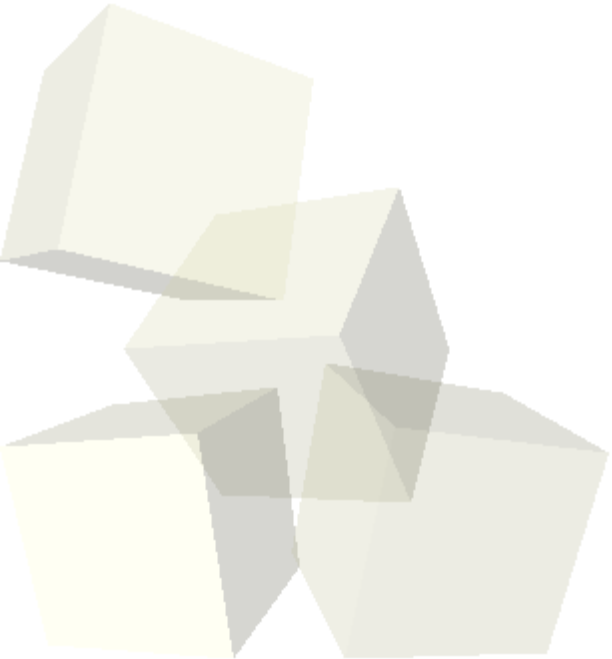
Complex Type: Sequence

```
<complexType>
  <sequence>
    <element name="tag1" type="s:integer" />
    <element name="tag2" type="s:integer" />
    <element name="tag3" type="s:integer" />
  </sequence>
</complexType>
```

A → *BCD*
B → <tag1> *E* </tag1>
C → <tag2> *F* </tag2>
D → <tag3> *G* </tag3>

Complex Type: Choice

```
<complexType>
  <choice>
    <element name="tag1" type="s:integer" />
    <element name="tag2" type="s:integer" />
    <element name="tag3" type="s:integer" />
  </choice>
</complexType>
```



A → *B*
A → *C*
A → *D*
B → <tag1> *E* </tag1>
C → <tag2> *F* </tag2>
D → <tag3> *G* </tag3>

Complex Type: All

```
<complexType>
  <all>
    <element name="tag1" type="s:integer" />
    <element name="tag2" type="s:integer" />
    <element name="tag3" type="s:integer" />
  </all>
</complexType>
```

$$A \rightarrow BCD$$
$$A \rightarrow BDC$$
$$A \rightarrow CBD$$
$$A \rightarrow CDB$$
$$A \rightarrow DBC$$
$$A \rightarrow DCB$$

XML Message

```
<status>
  <fire>false</fire>
  <light>4</light>
  <temp>23.4</temp>
  <temp>18.3</temp>
  <temp>20.7</temp>
  <client>mitra</client>
  <client>spectra</client>
   ...
</status>
```

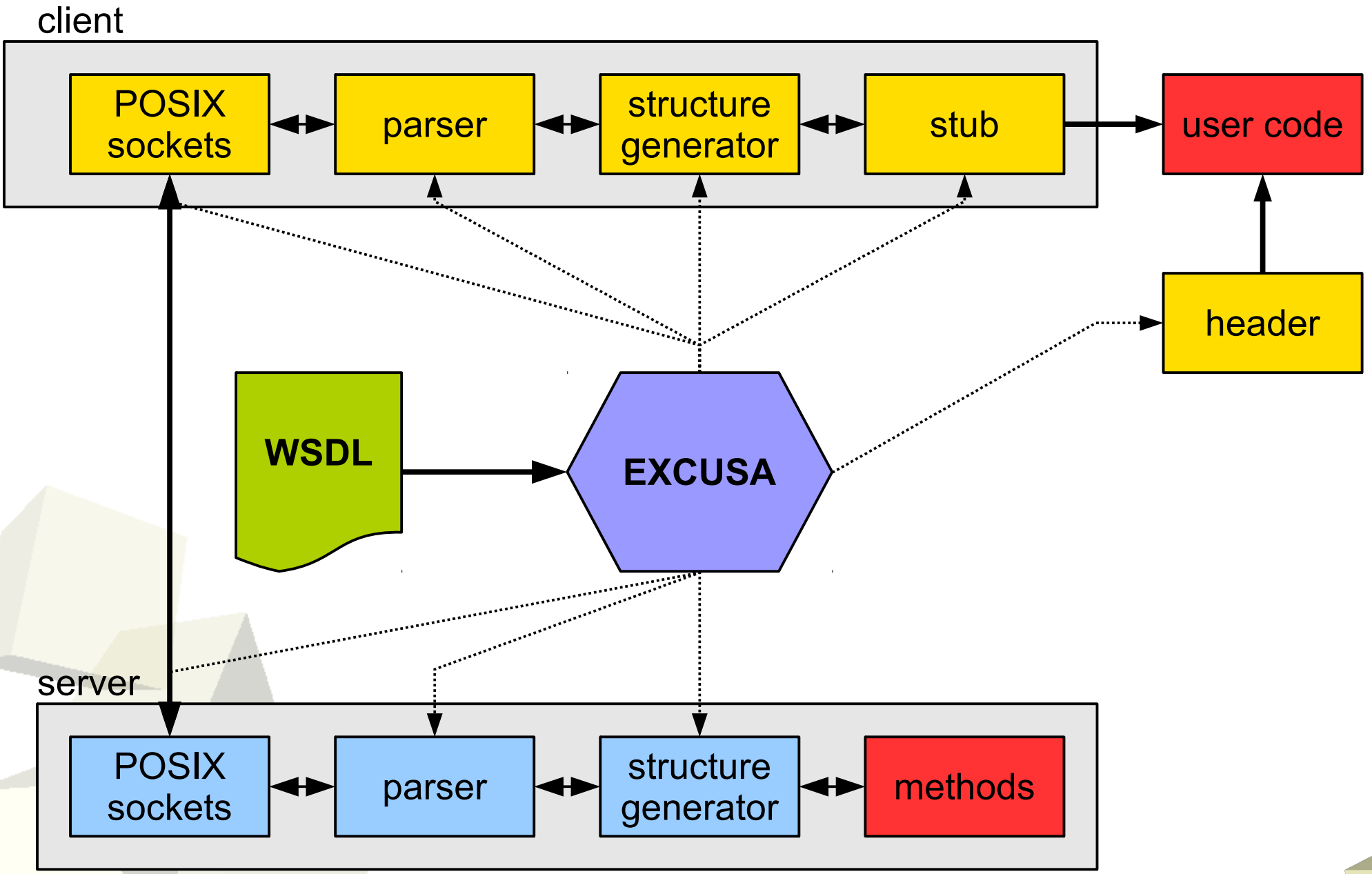
XML Grammar

S → <status> **A** </status>
A → **BDFI**
B → <fire> **C** </fire>
C → *{boolean}*
D → <light> **E** </light>
E → *{integer}*
F → **GGG**
G → <temp> **H** </temp>
H → *{float}*
I → **IJ**
J → ε
K → <client> **L** </client>
L → *{string}*

rule types:

- tag
- substitution
- value type

EXCUSA Schema



Discussion

Thanks!

http://github.com/prusnak/excusa