one offers solutions for tricky cases to encompass any deficiency that might appear in both SCXML and UML.

2.5 Discovering State Chart XML

This section singles out the learning of Statecharts in an intuitive manner. Accordingly, we draw statecharts (a.k.a. state machine diagrams or models) with the help of the UML formalism (SCXML has no proper graphical formalism). We also express statecharts by means of the SCXML dialect, which simply instantiates a given XML Schema Definition (XSD), i.e., a XML derivative. Any SCXML "model" or "document," as any phrase in a given language, then obeys construction rules established by this XSD. Current XSD is available at http://www.w3.org/2011/04/SCXML/scxml.xsd.

In this didactical section, in order to facilitate comprehension, we redesign as a statechart the notion of stack data structure. In absolute terms, there is no value to express a stack software component with Statecharts. This example is then just a toy example, but it has the great advantage of being highly pedagogical because everybody knows what a stack is and how it behaves.

2.5.1 "My stack" as Statechart Example

In Statecharts, from a structural viewpoint, state relationship types are:

- exclusiveness,
- orthogonality (a.k.a. parallelism or concurrency), and
- nesting.

Abstracting a stack data structure as a statechart may then lead to four states: "Empty," "Not empty," "Only one," and "More than one." In Figure 2.6, "My stack" is the name of a state machine diagram being the UML behavioral specification of a stack data structure. Naturally, "Empty" and "Not empty" are exclusive states. Besides, "Only one" and "More than one" are sub-cases of "Not empty," i.e., they are its substates. There is no use of the orthogonality relationship in this example.

In SCXML, nesting relationships are expressed by enclosing `<state>` tags in other `<state>` tags:

```
<state id="Not_empty">
   <state id="Only_one"> . . . </state>
   <state id="More_than_one"> . . . </state>
</state>
```