Statistics Netherlands

Division Research and Development
Department of Statistical Methods

P.O.Box 4000
2270 JM Voorburg
The Netherlands

The TADEQ Project, State of Affairs

Jelke Bethlehem and Anco Hundepool

Project number: *RSM-*
BPA number: *-RSM*
Date: *24 March 2000*

THE TADEQ PROJECT, STATE OF AFFAIRS

Summary: National Statistical Institutes, research institutes, and commercial marketing research organisations are more and more using computer-assisted interviewing (CAI) systems for collecting survey data. The growing possibilities of computer hardware and software have made it possible to develop very large, and complex electronic questionnaires. Unfortunately, it also has become more and more difficult for developers, interviewers, supervisors, and managers to keep control of the content and structure of CAI instruments. The TADEQ Project aims at developing a tool to make a readable and understandable documentation of an electronic questionnaire. This contribution presents an short overview of the project, and describes the current state of affairs.

Keywords: Electronic questionnaires, Documentation, XML

1. Introduction

National Statistical Institutes, research institutes, and commercial marketing research organisations are more and more using computer-assisted interviewing (CAI) systems for collecting survey data. They replace paper questionnaires by a computer program that guides respondents through the questionnaire and checks the answers on the spot. The growing possibilities of computer hardware and software have made it possible to develop very large, and complex electronic questionnaires. Unfortunately, it also has become more and more difficult for developers, interviewers, supervisors, and managers to keep control of the content and structure of CAI instruments.

The TADEQ Project aims at developing a tool to make a readable and understandable presentation of an electronic questionnaire. TADEQ stands for Tool for the Analysis and Documentation of Electronic Questionnaires. It is a Fourth Framework Research Project of the European Union. Institutes from five different countries co-operate in this project: Statistics Netherlands, the Technical University of Vienna (Austria), the Office for National Statistics (UK), Statistics Finland, and the Instituto Nacional de Estatística (Portugal).

TADEQ will be an open tool. It will offer facilities to create interfaces to various computer assisted interviewing systems. However, the focus is on Blaise. The open approach requires a neutral way to describe how an electronic questionnaire is executed. For this purpose the Questionnaire Documentation Language (QDL) has been developed. It is based on XML.

The documentation tool must be able to produce human-readable documentation both in paper and electronic form. On the one hand, this tool must be able to show

the global structure of the questionnaire, and on the other, it must provide means to focus on the details of parts of the questionnaire. A particular challenge of the TADEQ project is to display the routing graph of large and complex questionnaires. Due to the limited size of a sheet of paper and a computer screen, this is not a simple task. It must be accomplished without affecting the readability, so a lot of attention must be paid to layout issues.

Questionnaire documentation will be used by different people involved in the survey process. Examples are the questionnaire developer, who wants to document his work, the survey manager who has to give a formal approval for carrying out the survey, and the interviewers, who want paper documentation of the questionnaire to help them in the preparation and execution of their fieldwork. Different users mean different formats of the questionnaire documentation. So, a proposed documentation tool must be flexible. Users of this tool must have some control on adjusting the documentation. Research must be used to show what is required by whom.

The TADEQ project aims at developing a documentation tool that is at least capable of generating two types of documentation. In the first place, there will be textual documentation. It focuses on giving detailed information on all questionnaire objects (questions, checks, computations, etc). Routing information will be taken care of by attaching a condition to each questionnaire object. The questionnaire object will only be executed in situations in which the conditions are satisfied. A good example of this approach is the BAD system developed by the Office for National Statistics in the United Kingdom, see Andersen (1997).

The second type of documentation to be produced by TADEQ is a representation of the routing structure in graphical format. This type of documentation focuses more on the routing structure, and less on the details of the questionnaire objects. A limited amount of textual information can be displayed in the graph. Depending on the CAI system used, there is a choice: question identification names/numbers, question text (possibly in different languages), specification of the type of accepted answers, etc. The available amount of space in the graph is too limited to display all this information. Moreover it might affect readability. Therefore, a documentation tool must provide means to select the information shown, and possibly also means to display information in different ways, e.g. adjacent to the graph.

Some more information about the TADEQ project can be found in Bethlehem (1999).

2. The Questionnaire Documentation Language

To be able to describe what can happen during the execution of an electronic questionnaire, it is important to distinguish the different types of events that may occur and the conditions under which they occur. Examples of events are:

- Asking a question;

- Checking a relationship;
- Carrying out a computation.

The conditions under which these events happen are defined in the routing structure of the questionnaire. Depending on the type of computer assisted interviewing system, two approaches can be observed:

- GOTO oriented routing. These routing structures can either be unconditional (if attached to the answers to closed questions) or conditional (if attached to the outcome of logical expression). An example of such a CAI system is CASES.
- IF-THEN-ELSE oriented routing. This is more structural approach. The execution of blocks of questions depends on the value of a logical expression. An example of such a CAI system is Blaise.

The Questionnaire Definition Language (QDL) must be capable to cope with all possible events and routing structures. QDL is based on XML (Extended Markup Language). XML can be seen as the future successor of HTML, the language that is used to make web sites. HTML has the disadvantage that it is a mix of describing structure and layout. Moreover, the language is not extensible. XML is much more powerful. Users can define their own language elements, and structure is separated from layout. It is becoming more and more clear that XML is not only a powerful language for designing web sites, but that it is also very useful for defining the structure of data files. See also Boumphrey et al. (1998), or Morrison et al. (2000).

The XML language allows for the definition of the different events one might encounter in the execution of an electronic questionnaire. In QDL such events are called questionnaire object. Examples of questionnaire object are questions (various types), checks, computations, route instructions, sub-questionnaires, etc. Each object has a number of attributes, like a question text (for questions), logical expressions (for route instructions and checks), and arithmetical expressions (for computations).

TADEQ expects the questionnaire definition to be in QDL format. Because QDL is an XML application, the information can be processed in various ways. One way is to make use of an XML parser offered by Microsoft. This parser comes for free with the Internet Explorer 5.0 browser. It is a DLL-file (called MSXML.DLL), which can be used in a C++, Delphi, or Visual Basic program.

For textual documentation, another way is to make use of *style sheets*. They offer a means of defining layout for an XML document. One way of doing this is using XSL (Extended Stylesheet Language). The advantage of a style sheet is that it is fairly simple for users to define their own layout formats. However, style sheets have their limitations. The approach of creating a dedicated layout module using the XML parser offers more possibilities. TADEQ will only offer some basic examples of XSL style sheets. Of course, users can always create their own XSL style sheets.

To illustrate how TADEQ works, we will use a simple example of a Blaise questionnaire. The data model for this example is presented in figure 2.1. It contains a number of nested IF-statements and a check.

Figure 2.1. The Blaise data model

```
DATAMODEL Commuter "The Commuting Survey"

FIELDS
  Sex        "What is your sex?": (Male, Female)
  Age        "What is your age?": 0..120
  MarStat    "What is your marital status?":
             (NeverMar "Never married",
              Married  "Married",
              Divorced "Divorced",
              Widowed  "Widowed")
  Work       "Do you have a paid job?": (Yes, No)
  DistWork   "What is the distance to work?": 0..300
  TypeWork   "What type of work do you have?": string[40]
  LookWork   "Are you looking for a job?": (Yes, No)
  School     "Are you still going to school?": (Yes, No)
  DistSchool "What is the distance to school?": 0..300
  Activity   "What is your main activity?": string[40]

 RULES
   Sex Age MarStat
   IF Age < 15 THEN
     School
     IF School = Yes THEN
       DistSchool
     ELSE
       Activity
     ENDIF
     MarStat = NeverMar
   ELSE
     Work
     IF Work = No THEN
       LookWork
     ELSE
       TypeWork DistWork
     ENDIF
   ENDIF

ENDMODEL
```

To be able to process the metadata, TADEQ must translate the data model into QDL. It should be realised that QDL not meant to mimic the Blaise language or the authoring language of any other CAI system. The objective of QDL to describe what can happen during the execution of an electronic questionnaire. So it is a description of the events that can happen and the conditions under which they can happen.

The basic building block of QDL is the Questionnaire Object. Examples of Questionnaire Objects are question objects (various types), navigation objects (IF-THEN-ELSE, GOTO, Loop), computation objects, check objects, etc.

Figure 2.2 contains an example of a question object. It is a closed question with two possible answers. Each Questionnaire Object has a unique name and sequence number. In this example there is only one question text, but QDL allows for more than one. The attribute `max` of the `<closed>` tag indicates how many answers are allowed. For each possible answer the specification contains an `<item>` tag. Note the use of CDATA to specify plain text. This prevents special symbols in the text to be interpreted as XML tags.

Figure 2.2. A question object in QDL

```
<qobject number="1" name="Sex">
  <question>
    <text><![CDATA[What is your sex?]]></text>
    <closed max="1">
      <item code="1" name="Male">
        <text><![CDATA[Male]]></text>
      </item>
      <item code="2" name="Female">
        <text><![CDATA[Female]]></text>
      </item>
    </closed>
    <status><![CDATA[ask]]></status>
  </question>
</qobject>
```

Figure 2.3 shows how an IF-THEN-ELSE structure is specified in QDL. Such a structure is identified by means of a `<split>` tag. The logical expression to be evaluated can be found between the `<condition>` tags. If the expression turns out to be true, part of the questionnaire between the `<if_true>` tags is executed, and if it is false, the part between the `<if_false>` tags is carried out. Note that a `<text>` tag will be included after the `<condition>` tag if an explanatory text is added to the condition.

Figure 2.3. An IF-THEN-ELSE structure in QDL.

```
<qobject number="4" name="Condition">
  <split>
    <condition><![CDATA[Age < 15]]></condition>
    <if_true>
      <qobject number="5" name="School">
        <question>
        . . .
        </question>
      </qobject>
      . . .
    </if_true>
    <if_false>
      <qobject number="10" name="Work">
        <question>
        . . .
        </question>
      </qobject>
      . . .
    </if_false>
   </split>
</qobject>
```

A final example of a QDL object is the check specified in figure 2.4. The attribute `type` of the `<check>` tag indicates whether this a hard check or a soft check (signal). Although this example does not show it, a list of involved questions can be included in the specification. Also an explanatory text can be included.

Figure 2.4. A check in QDL

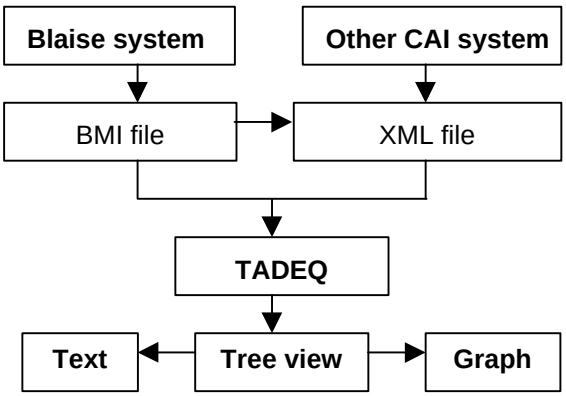
```
<qobject number="9" name="Check">
   <check type="hard">
     <condition><![CDATA[MarStat = NeverMar]]></condition>
   </check>
</qobject>
```

The questionnaire specification in QDL forms the input for TADEQ. The subsequent sections describe what TADEQ can do with this input.

3. The architecture of TADEQ

TADEQ must be able to produce textual and graphical documentation both in electronic and paper form. Starting point always is the electronic version of the documentation. The users can interactively set display and layout settings, and fold or unfold sub-questionnaires. If they are satisfied with the result, they can instruct the program to print the documentation. Figure 3.1 below summarises the current architecture of TADEQ prototype.

Figure 3.1. The architecture of the TADEQ prototype



TADEQ accepts two types of input:

- A Blaise metadata file. For the Blaise system, there is a direct link between TADEQ and Blaise. TADEQ is able to read a Blaise metadata file (a so-called BMI file). Internally, TADEQ converts the BMI file into a QDL file using the Blaise API.
- A questionnaire specification in the form of a QDL document. This is an XML document satisfying the syntax rules specified in the TADEQ Data Type Definition (DTD).

The second type of input will serve all CAI packages, with the exception of Blaise. It means that for all these packages a conversion tool must be offered capable of

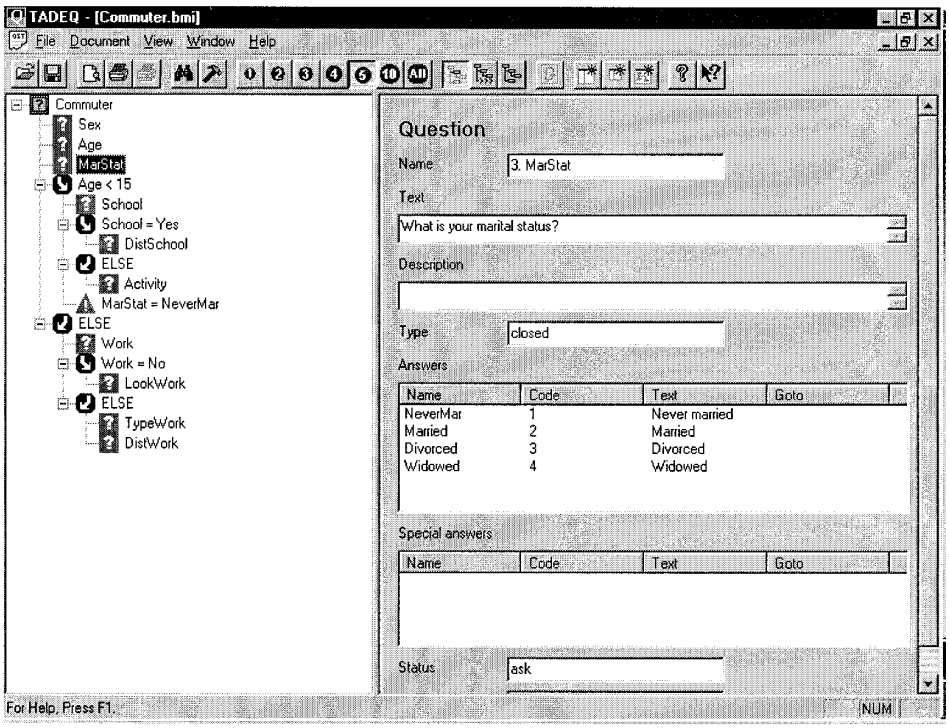
transforming the dedicated questionnaire definition format into an QDL document. This will be the responsibility of the developers of these packages.

Whatever the source of the information (Blaise or another CAI package), TADEQ will store the information always in an XML-document. Consequently, all parts of TADEQ producing output are able to read XML.

Microsoft has made available a DLL containing routines for parsing and processing XML files. The name of this file is MSXML.DLL, and it is available for free in the Internet Explorer Version 5 environment. This DLL can be used in a C++, Delphi, Visual Basic, Java, or Java Script program. TADEQ has been written in C++.

After reading in a QDL file, or reading a Blaise BMI file and transforming it into an QDL file, TADEQ will present the information on the screen in the form of a tree. See the left hand side of figure 3.2 for an example.

Figure 3.2. TADEQ screen with the questionnaire tree



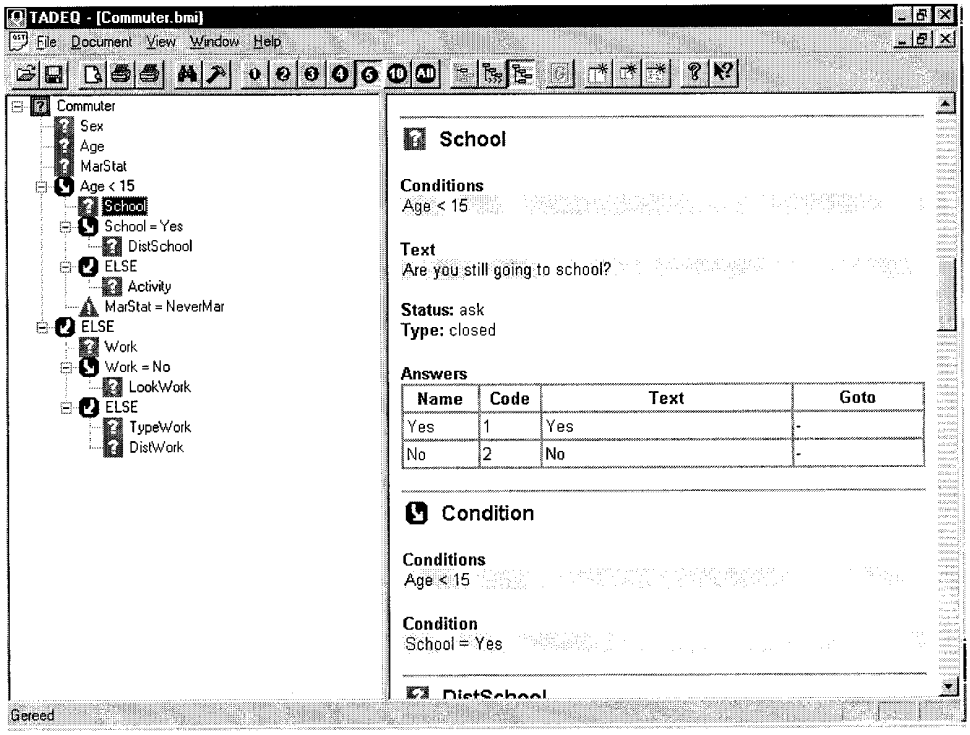
Each questionnaire object is represented by a small icon and a name. The various branches of the execution tree are indicated by means of indentation. Sub-branches may be *collapsed* (i.e. closed so that they are temporarily invisible) or *expanded* (i.e. opened so that the become visible). The tree can be used to navigate through the questionnaire, expand or collapse branches, and focus on specific parts of the questionnaire.

For the right hand side of the screen, the user can make a choice between two alternatives. The first one is *object information panel*. This panel displays all details of the object which has the focus in the tree on the left hand side. The right hand side

of figure 3.2 shows the information displayed for the question object MarStat. All tag and attribute texts of the question object can be found by scrolling through the panel.

An alternative for the right hand side of the screen is the complete *questionnaire documentation panel*. A choice for this panel will produce complete questionnaire documentation in HTML format. See the right hand side of figure 3.3 for an example.

Figure 3.3. TADEQ screen with HTML documentation



The user can scroll through this panel to see other parts of the document. If a questionnaire object contains a reference to another questionnaire object, the user can jump to that object by means of hypertext links.

Each object description starts with a list of conditions under which it will be executed. For example, the question School in figure 3.3 is only asked of people of age less than 15.

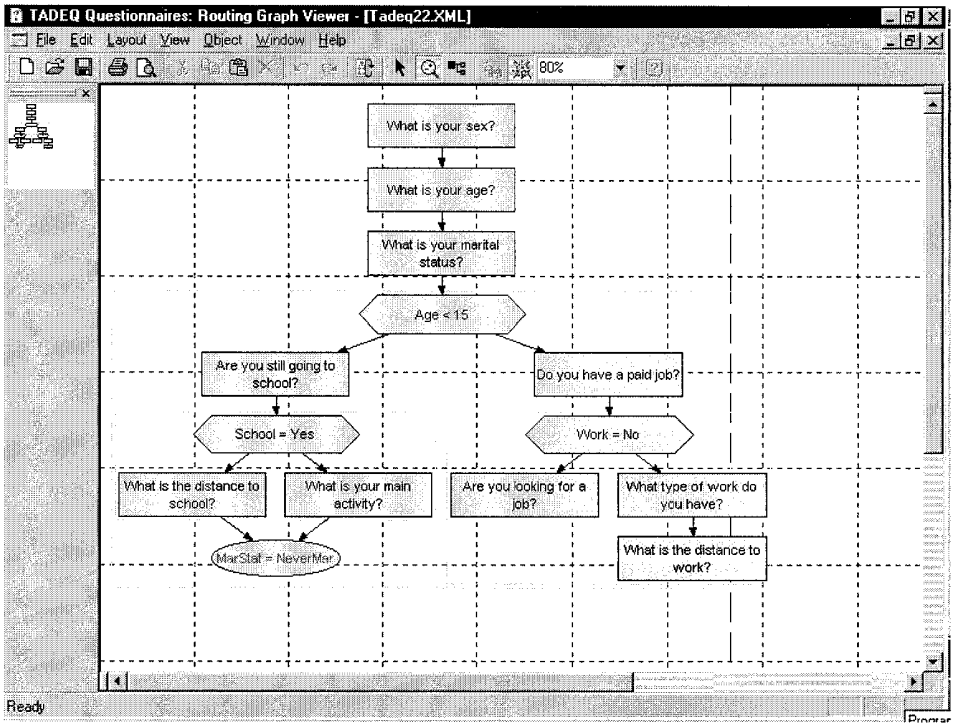
TADEQ has numerous display options. The tool supports multilingual questionnaire, so one can set the language of the documentation. It is also possible to set a filter controlling the type of object to be displayed (question, check, computation, etc). Another option allows the user to see only the last condition leading to the current object, or to display the complete list of conditions from the start of the questionnaire leading to this object.

The HTML documentation can be saved in a file for later use, like e.g. printing. The layout of the documentation is determined by a Cascading Style Sheet (CSS). By editing the CSS file, the user can change the appearance of the information in the files (font, font size, font colour, etc).

The HTML output of TADEQ can be read by MS Word. Once available in this word processor, the information can be easily turned into printable documentation. Note that MS Word sees object names as section headings. This allows for the creation of a table of contents with one mouse click.

To get more insight in the routing structure of the questionnaire, TADEQ can generate graphical output. For this it makes a call to a graph drawing routine (QDRAW). This routine uses the QDL file for input. Figure 3.4 shows an example of such a graph.

Figure 3.4. TADEQ screen with graphical output



The rectangle denotes a question. The hexagonal shape represents an IF-THEN-ELSE object where the left branch is followed if the condition is true and the right one if the condition is false. The oval shape denotes a check object.

The user will have a choice for displaying two types of information:

- A more technical and compact representation uses question names and mathematical expressions in the flowchart symbols;
- A more non-technical and readable representation will use question texts and explanatory texts in the flowchart symbols.

Like in the tree view, one can by simple clicking collapse or expand branches in this flow chart.

For this simple example, the graph fits on the screen. For larger questionnaires one can scroll through the graph. The small navigation in the upper left corner of the screen gives the user the possibility to quickly move to a different part of the graph.

The graph can be printed. Since the size of a sheet of paper is finite, printing poses special problems. Algorithms will be implemented in TADEQ to divide the graph over a number of sheets in a neat way, without cutting too many lines. Also, special reference symbols will be included, so that users can quickly find the spot on another sheet where a specific line is continued.

4. Future developments

TADEQ is still in development. One of the things that will be added is a number of analysis functions. With these functions one can obtain more insight in the structure of the routing. Several types of functions are considered.

There will be basic statistics, like frequencies of occurrence of the various types of questionnaire objects. Also, there probably will be statistics on the (weighted or unweighted) lengths of the various paths through the questionnaire. If the weight is the expected time to ask and answer a question, this may provide information about expected duration of interviews, and the variation in interviewing length.

More thorough analysis is possible if information is available on the frequencies with which questionnaire objects are encountered in interviews. Such information can be generated in two ways: (1) before the field work starts by randomly generating interview data that satisfy the routing conditions, and (2) after the fieldwork has been completed by processing the data file. Research is been carried out whether it is possible in a simple way to include such information in the QDL file. It implies every questionnaire will get an extra `<freq>` tag. It is probably not too difficult to fill the this tag for question objects, but more interesting analysis can be carried out if this tag can be filled for the decision objects (IF-THEN-ELSE, GOTO).

Another interesting function could be to evaluate all expressions encountered on a path through the questionnaire. This gives information about the characteristics of persons following this path. And it may even turn out that no one will be able to follow this path, because conditions contradict one another. Thus, this type of information would help to develop more balanced questionnaires. Unfortunately, this asks for developing a complete expression evaluator, which comes more or less down to re-building Blaise. One may wonder whether this is worth the effort. Another complication is that some expressions are impossible to evaluate because information is required that is not available at the time of testing (like values from external files).

Another the analyse the routing structure could be to reduce questionnaire tree to only those objects that play a role in routing decisions. This would result in only displaying the decision objects and the questions determining the value of the routing conditions.

5. References

Anderson, S. (1997), Automated Paper Documentation of Blaise III. Actes de la 4e Conférence Internationale des Utilisateurs de BLAISE, INSEE, Paris, pp. 1-20.

Bethlehem, J.G. (1999): The Routing Structure of Questionnaires. Proceedings of the Third ASC International Conference, Association of Survey Computing, Chesham, United Kingdom, pp. 405-418.

Boumphrey, F., Direnzo, O., Duckett, J, Graf, J., Hollander, D., Houle, P., Jenkins, T., Jones, P., Kingsley-Hughes, A., Kingsley-Hughes, K., McQueen, C., and Mohr, S. (1998): XML Applications. Wrox Press, Birmingham, UK.

Morrison, M., Boumphrey, F. and Brownell, D. (2000): XML Unleashed. Sams Publishing, Indianapols, USA.