Production Scheduling Concepts Modeling through XML

Leonilde Varela[1], Joaquim Aparício[2], Carmo Silva[3]

Dept. Production Systems, University of Minho,
[1] Azurém Campus, 4800-058 Guimarães, Portugal
[3] Gualtar Campus, 4710-057 Braga, Portugal
{Leonilde, Scarmo}@dps.uminho.pt
http://www.dps.uminho.pt/pessoais
[2] Dept. Computer Science, New University of Lisbon,
Quinta da Torre, 2829-516, Monte de Caparica, Portugal
Jna@di.fct.unl.pt
http://www.di.fct.unl.pt/

Abstract. Production scheduling objectives can be better satisfied through the execution of the most suitable methods available for the resolution of each particular problem. In this paper we make a contribution to this through scheduling knowledge modelling and communication through XML (eXtensible Markup Language) and related technologies based on a production scheduling problems classification nomenclature developed. We aim at standard representation of scheduling problems, methods and related concepts, including standard access to scheduling methods implementations through the Internet. This kind of data modelling is the support for a web-based decision support system for solving such problems. Such a system has the purpose of allowing different users to easily access, share and disseminate knowledge about production scheduling through the Internet.

1 Introduction

Scheduling problems are a part of a much broader class of combinatorial optimisation problems. Good production schedules strongly contribute to the increase of a companies' success. The objective in a production scheduling problem consists on finding a solution for allocating a set of machines to process a set of jobs in a given time horizon in order to obtain good system performance based on single or multi criteria.

Scheduling problems are characterized by several constrains, namely job ready times, due dates, penalties for delays, and many other constraints related to task processing and production resources. These include job operations precedence constraints and sometimes precedence constraints among jobs. Typical examples of performance criteria include the minimization of maximum job flow and maximum lateness of jobs in the production system.

The wide spectrum of scheduling problems calls for a standardized nomenclature for problem representation, based on attributes that enable defining problem classes to which real problem instances belong. Such a nomenclature is important not only for problem representation but also for problem resolution. In our work, we developed a nomenclature for such purpose based on work presented in some important and well-known literature sources on scheduling [3,4,13,15,17].

The scheduling objectives can be better satisfied through the execution of the most suitable methods available for the resolution of each particular problem. In this paper we make a contribution to this through scheduling knowledge modelling and communication through XML (eXtensible Markup Language) and related technologies based on the classification nomenclature mentioned. We aim at standard representation of scheduling problems, methods and other scheduling concepts, including standard access to scheduling methods implementations through the Internet. This kind of data modelling is the support for a web-based decision support system for solving such problems. Such a system has the purpose of allowing different users to easily access, share and disseminate knowledge about production scheduling through the Internet.

This paper is organized as follows. The next section briefly describes the nature of production scheduling problems and the classification model used. Section 3 presents the XML-based modeling of the main production scheduling concepts, which include the problems, the methods and the results specification. Section 4 briefly refers to the remote methods invocation process, illustrated through the XML-RPC communication protocol, and finally, in section 5 we present the conclusion.

2 Production Scheduling

Production Scheduling problems have a set of characteristics that need specification. These characteristics can be organized into classes. One such class of factors, which we call the α class, characterizes the production environment, i.e. the system and machines available. Another one, the β class, deals mainly with characterization of jobs and processing requirements. Some important processing requirements that frequently have to be taken into account for processing jobs have to do with resources other than machines, i.e. operators, tools, handling devices buffers among others. These must also be specified and are considered in our nomenclature for problem definition [19,20]. The third, the γ class, specifies the performance measure or evaluation criterion. Typical examples of such measures are the maximum flow time, the makespan and the mean and maximum lateness of jobs.

The classification nomenclature is used as a basis for the XML-based problem specification model underlying this work. It includes a wide range of problem classification factors, which may be combined in different ways, resulting in many distinct scheduling problem classes. For example, class F2|n|Cmax, refers to the problem of processing n jobs on a pure flow shop, with two machines, always available, with jobs being ready at time zero for processing. The performance measure consists on minimizing the maximum completion time or makespan (Cmax).

3 XML Modeling

The eXtensible Markup Language (XML) has been having a wide acceptance and caused a great impact on Internet real world applications, since its release by the World Wide Web Consortium (W3C) in 1998 [16]. This specification language enables describing structures and meanings of data, with a simple syntax, and is an ideal candidate format for exchanging and processing data through the Internet. Other advantages of XML based representation are its openness, simplicity and scalability [2,6,11,16,18]. These were some of the most important reasons for having chosen it for the development of our web system. For details about XML and related technologies (DTD, XSL, XML Schemas, Namespaces, etc.) see, for example, Ceponkus and Hoodbhoy [6] or Ramalho and Henriques [18], among many other references [2,5,6,16].

3.1 Problems

Following the lines already presented in [19,20] problems are classified and modelled by a DTD (Document Type Definition) (see code below). Elements introduced in the referred DTD are expected to become part of a common namespace. Elements on the problem DTD file precisely characterize a scheduling problem class, meaning that from the point of view of the system and interaction with the system a problem must be described according to that grammar.

```
Elements and attributes declaration -->
<!ELEMENT problems (problem+)>
<!ELEMENT problem (alpha?, beta?, gamma)>
<!ATTLIST problem
 problem_class CDATA #REQUIRED
 preferred (true | false) "false">
<!-- Alpha elements -->
<!ELEMENT alpha (alpha1?,alpha2?)>
<!-- Alpha1 -->
<!ELEMENT alpha1 EMPTY>
<!ATTLIST alpha1 system_type (0 | P | PMPM |…) "0">
<!-- Alpha2 -->
<!ELEMENT alpha2 EMPTY>
<!ATTLIST alpha2
 value (0 | m) "0"
    machines_quantity CDATA #REQUIRED>
<!-- Beta elements -->
<!ELEMENT beta (beta1?, beta2?, ... , beta12?)>
…
<!-- Beta4 -->
<!ELEMENT beta4 (job_time*)>
<!ATTLIST beta4 value (0 | pj) "0">
<!ELEMENT job_time EMPTY>
<!ATTLIST job_time
    name CDATA #REQUIRED
    time CDATA #REQUIRED>
```

```
  <!-- Beta5 -->
  <!ELEMENT beta5 (job_date*)>
  <!ATTLIST beta5 value (0 | dj) "0">
  <!ELEMENT job_date EMPTY>
  <!ATTLIST job_date
     name CDATA #REQUIRED
     date CDATA #REQUIRED>
  …
  <!-- Beta7 -->
  <!ELEMENT beta7 EMPTY>
  <!ATTLIST beta7 value (0 | nj) "0"
   jobs_quantity CDATA #REQUIRED>
  <!—Beta8 -->
  <!ELEMENT beta8 (job_priority*)>
  <!ATTLIST beta8 value (0 | dj) "0">
  <!ELEMENT job_priority EMPTY>
  <!ATTLIST job_priority
     name CDATA #REQUIRED
     priority CDATA #REQUIRED>
  …
  <!-- Gamma element -->
  <!ELEMENT gamma EMPTY>
  <!ATTLIST gamma measure (Cmax | SumCj | Cmean | SumWjCj
| Fmax | SumFj | Fmean | SumWjFj | Lmax | SumLj |  Lmean
| SumWjLj | Tmax | SumTj | Tmean | SumWjTj | Emax |
SumEj...) “Cmax”>
```

From the code above, we can read that in order to define a problem we must optionally defined the alpha, beta and the gamma factors since there is always a corresponding factor value defined by default.

Our sample problem is known to belong to class F2|n|Cmax, and can be defined by an XML file as follows.

```
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE problems SYSTEM "problem.dtd">
  <problems>
      <problem problem_class="F2|n|Cmax">
          <alpha>
              <alpha1 system_type ="F"/>
              <alpha2 machines_quantity="2"/>
          </alpha>
          <beta>
             …
           <beta4 value="0">
        <job_time name="J1" time="10"/>
          </beta4>
          …
          <beta7 value="n" jobs_quantity="4"/>
          …
          </beta>
          <gamma measure="Cmax"/>
      </problem>
  </problems>
```

3.2 Methods

In the Internet many implementations may exist for a same method. From the point of view of the system two implementations of a given method may differ if, for example, they differ only on its outputs. Unfortunately not all implementations work in the same way and in order for the system to use those implementations in a programmatic way, they must also be described within the system. This description must include (among other things) the actual call to the running method. Implementations are described in a DTD file, and an example for the implementation for the Johnson's rule is given below.

```
<!ELEMENT methods (method)*>
<!ELEMENT
method(id,name,url?,problem_class,method_class?,referen
ce,complexity?,protocol?,signature?,gantt?)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT problem_class (#PCDATA)>
<!ELEMENT method_class (#PCDATA)>
<!ELEMENT reference (#PCDATA)>
<!ELEMENT complexity (#PCDATA)>
<!ELEMENT protocol (#PCDATA)>
<!ELEMENT signature (input,output)>
<!ELEMENT input (param | array | matrix)+>
<!ELEMENT param (#PCDATA)>
<!ATTLIST  param  name  CDATA  #REQUIRED  type  CDATA
#REQUIRED control (submit) #IMPLIED>
<!ELEMENT array (item+)>
<!ATTLIST  array  from  CDATA  #FIXED  “1”  to  CDATA
#REQUIRED control (submit) #IMPLIED>
<!ELEMENT item (#PCDATA)>
<!ATTLIST  item  name  CDATA  #REQUIRED  type  CDATA
#REQUIRED>
<!ELEMENT matrix (item+)>
<!ATTLIST  matrix  lines  CDATA #REQUIRED columns  CDATA
#REQUIRED control (submit) #IMPLIED>
<!ELEMENT output (param | array | matrix)+>
<!ELEMENT gantt (#PCDATA)>
```

Many scheduling methods may be more or less adequate to solve a given class of problems. In the methods knowledge base the system records the scheduling method(s) that can be used for solving a certain problem class. Searching for the adequate methods, for a given problem, is performed by matching the problem details with the methods' characteristics, a process performed by the built-in prolog engine. The methods can be available in the methods' knowledge base but they can also be found in any other sites, made available and accessible through the Internet.

The example code below shows a sample of the XML document about scheduling methods. It illustrates the information related to the implementation of the Johnson's Rule [1,3,4,8,9,10,12,15,17] for solving problem instances belonging to the

F2|n|Cmax class described in section 2. This document is validated against the corresponding DTD, previously shown, before being put in the corresponding knowledge base component.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE methods SYSTEM “methods.dtd”>
<methods>
 <method>

 <id>32</id><name>Johnson</name><url>http://localhost:60
02/RPC2</url>
   <prob_class>F2|n|Cmax</prob_class>
   <method_class>Sequencing Rule</method_class>
   <reference>Johnson,1954</reference>
   <complexity>Maximal         Polynomially         Sol
able</complexity>
<protocol>XML-RPC</protocol>
   <signature>
     <input>
       <param name=”n” type=”integer”/>
       <param     name=”m”     type=”integer”     con-
trol=”submit”/>
       <matrix lines=”m” columns=”n”>
         <item     name=”job”     type=”string”/><item
name=”machine” type=”string”/>
         <item name=”p” type=”double”/>
       </matrix>
     </input>
     <output>
       <param name=”Cmax” type=”double”/>
       <param name=”sequence” type=”string”/>
       …
       <matrix lines=”m” columns=”n”>
         <item name=”start” type=”double”/>
         <item name=”finish” type=”double”/>
         …
       </matrix>
     </output>
   </signature>
   <gantt>yes</gantt>
 </method>…
</methods>
```

The inputs include the definition of a parameter n, for the number of jobs to be processed, a parameter m, for the number of machines, and a set of three items organized as a matrix structure, which represent the job name, the machine name and the processing time p of each job on each machine. There is also the definition for the method’s output following the same lines. After a method definition has been inserted in the corresponding KB it becomes immediately accessible to any further information retrieval. For the example given, after the insertion of the Johnson’s method definition any search for methods that is a match for the F2|n|Cmax problem class will include this method in the search results. The methods’ definitions are also used in the automatic generation of interfaces for methods invocation and corresponding inputs insertion and subsequently outputs presentation.

3.3 Results

The production scheduling problems' results may be defined in many different ways. Gantt charts are very typical and useful way or expressing these problem results.
Gantt charts are time-based diagrams, which can also automatically generated by the system, given the outputs provided by methods' invocation. This is easily achieved because the methods' output data is expressed in XML documents, enabling an easy way of outputs conversion into different desired problem results presentation forms, namely Gantt charts.
The code below illustrates the DTD for specifying the problem results through such Gantt charts.

```
<!ELEMENT gantt (problem*)>
<!ELEMENT problem (job+)>
<!ATTLIST problem
          id CDATA #REQUIRED
          class CDATA #REQUIRED>
<!ELEMENT job (machine+)>
<!ATTLIST job
          id CDATA #REQUIRED
          name CDATA #REQUIRED>
<!ELEMENT machine (start+,finish+)>
<!ATTLIST machine
          id CDATA #REQUIRED
          name CDATA #REQUIRED>
<!ELEMENT start (#PCDATA)>
<!ELEMENT finish (#PCDATA)>
```

A possible corresponding XML example, for representing a given problem instance is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gantt SYSTEM "gantt.dtd">
<gantt>
  <problems>
    <problem id=”0010” class="F3|n|Cmax">
        <results>
            <job id="J1" name=”Job1”/>
              <machine id="M1" name=”Machine1”/>
              <start>0</start>
              <finish>3</finish>
            </job>…
          </results>
    </problem>
  </problems>
</gantt>
```

These time-based charts continue to be largely used due to its expressive power, enabling a very easy way to comparing results obtained from the invocation of several different implemented methods available. Other alternatives for displaying those outputs are possible, including direct outputs presentation through XML documents.

4 Methods Invocation

The main purpose of this work consists on trying to improve the resolution of production scheduling problems. Therefore, we decided to develop a web system, based on XML modeling and related technologies.
The main element of the web system architecture is an interface component for introduction, validation, and transformation of production scheduling data. This interface is mainly controlled by DTD and XSL (eXtensible Stylesheet Language) documents stored in a knowledge base. The scheduling information is also stored in XML documents and these documents are verified using DTDs, before being put in the corresponding XML knowledge base. The XML and related documents may either be located on the server or on the client side. In this work the documents are stored on the server (e.g. XML, DTD, XSL and other documents) in order to achieve easy and efficient data transferring.
The system is being implemented as a web service and allows the execution of either local or remote scheduling methods, namely through the XML-RPC protocol.
The term Web Services has emerged as a general category for loosely coupled, dynamically connected web-based services and are a set of tools that let us build distributed applications on top of existing web infrastructures.
These services use XML to encode both the message wrapper and the content of the message body. As a result, the integration is completely independent of operating system, language or other middleware product used by each component participating in the service. The only fundamental requirement is that each component has the ability to process XML documents and that each node connected in a distributed system supports HTTP as a default transport layer.
These Web Services are most commonly used to invoke remote application services or methods using a Remote Procedure Call (RPC) interaction implemented using only XML messages [5].
The XML-RPC protocol consists on a possible communication protocol that can be used for remote methods invocation and can be defined as the sequence and structure of requests and responses required to invoke communications on a remote machine. Several other protocols that could also be used exist, namely SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery, and Integration of business for the web), WSDL (Web Services Description Language), or other well known, like CORBA (Common Object Request Broker Architecture), RMI (Remote Method Invocation) or DCOM (Distributed Component Object Model). The XML-RPC protocol is one possible approach that makes it easy for computers to call procedures on other computers [14] and is being used in this work for illustrating the remote scheduling methods invocation process.

The extensible markup language provides a vocabulary for describing remote procedure calls, which are then transmitted between computers using the Hyper Text Transfer Protocol (HTTP).

XML-RPC clients make procedure requests of XML-RPC servers, which return results to the XML-RPC clients. XML-RPC clients use the same HTTP facilities as web browser clients, and XML-RPC servers use the same HTTP facilities as web servers.

XML-RPC requires a minimal number of HTTP headers to be sent along with the XML method request. The code below shows an example that joins the headers and XML payload to form a complete XML-RPC request for solving a given problem belonging to the F2|n|Cmax problem class.

```
POST /rpchandler HTTP/1.0
User-Agent: AcmeXMLRPC/1.0
Host:localhost:5001
Content-Type: text/xml
Content-Length: 832
<?xml version="1.0"?>
<methodCall>
<methodName>JohnsonRule</methodName>
<params>
  <param><value><int>4</int></value></param>
  <param><value><int>2</int></value></param>
  <param>
   <value>
    <array>
      <data>
         <value><string>J1</string></value>
         <value><string>M1</string></value>
         <value><double>3</double></value>
         …
      </data>
    </array>
   </value>
  </param>
  </params>
</methodCall>
```

Upon receiving an XML-RPC request, an XML-RPC server must deliver a response to the client. The response may take one of two forms: the result of processing the method or a fault report, indicating that something has gone wrong in handling the request from the client. As with an XML-RPC request, the response consists of HTTP headers and an XML payload.

The code below shows a complete response from an XML-RPC server, including both the HTTP headers and the XML payload.

```
HTTP/1.0 927 OK
Date: Fri, 25 Oct 2002 07:38:05 GMT
Server: MyCustomXMLRPCserver
Connection: close
Content-Type: text/xml
Content-Length: 868
```

```
  <?xml version="1.0"?>
  <methodResponse>
   <params>
    <param>
     <value><string>L1,L3,L4,L2</string></value>
    </param>
    <param><value><double>39</double></value></param>
    <param>
     <value>
      <array>
        <data>
           <value><string>L1</string></value>
           <value><string>M1</string></value>
           <value><double>0</double></value>
           <value><double>3</double></value>
           …
       </data>
      </array>
     </value>
    </param>
 </params>
</methodResponse>
```

The response is provided to a call to the method JohnsonRule, which returns a solution to a F2|n|Cmax problem.

By using the XML-RPC protocol we are able to invoke scheduling methods implemented on different programming languages. Moreover, these methods, local or remotely available, may be running on different platforms.
As referred previously, our web scheduling system includes a Knowledge Base component that encompasses all the knowledge and modules necessary for remote methods invocation and for performing a set of other system functionalities. This component is controlled by ASP (Active Server Pages) and corresponding server-side XML-RPC components.
On the other hand, there are also correspondent XML-RPC components for each of the methods servers waiting for RPC requests.
This environment is heterogeneous as servers can use their own technology, i.e. use different implementation languages or/and different operating systems.
More detailed information about the XML-RPC protocol can be obtained from http://www.xmlrpc.com.

5 Conclusion

In production enterprises, it is important nowadays, as a competitive strategy, to explore and use software applications, now becoming available through the Internet and Intranets, for solving scheduling problems, which can be achieved in an easy way by using web service technology.
This work is based on an XML-based specification framework for production scheduling concepts modeling, which is used as specification framework for production

scheduling decision-making through a web service. Some of the important functions include the ability to represent scheduling problems and the identification of appropriate available methods for solving them.
The XML-based data modeling is used in order to make possible flexible communication among different scheduling applications. This modeling and specification contributes to the improvement of the scheduling process, by allowing an easy selection of several alternative methods available for problems solving, as well as an easy maintenance of the knowledge base supporting the scheduling service. This is achieved by providing a user-friendly way of new knowledge insertion. This knowledge primarily includes scheduling problems and solving methods as well as its implementations, available through the Internet. These may be implemented on different programming languages and running on different platforms. Such implementations are easily accessible through the web service referred in this paper for solving scheduling problems, through the invocation of local or remotely available scheduling methods. The system allows comparing different solutions obtained by running different methods for a same scheduling problem, and to choose the best solution found to solve the problem, according to a given performance measure to be reached.
The XML based specification can be generated and visualized by computers in appropriate and different ways. An important issue is that the data representation model is general, accommodating a large variety of scheduling problems, which may occur in different types of production environments.

References

1. Artiba, A., Elmaghraby, S.: The Planning and Scheduling of Production Systems. Chapman & Hall, UK (1997).
2. Abiteboul, S., et al.: Data on the Web - From Relations to Semistructured Data and XML. Morgan Kaufmann Publishers, USA (2000).
3. Blazewicz, J., et al.: Scheduling Computer and Manufacturing Processes. Springer-Verlag, Germany (1996).
4. Brucker, P.: Scheduling Algorithms. Springer-Verlag, Germany (1995).
5. Carlson, D.: Modeling XML Applications with UML – Practical e-Business Applications. Addison-Wesley, USA (2001).
6. Ceponkus, A., Hoodbhoy, F.: Applied XML. Wiley Computer Publishing, USA (1999).
7. Chrétienne, P., et al.: Scheduling Theory and its Applications. John Wiley & Sons Inc., England (1995).
8. Conway, R. W., Maxwell, W. L., Miller, L. W.: Theory of Scheduling. Addison-Wesley Publishing Company, Inc., England (1967).
9. French, S.: Sequencing and Scheduling – An Introduction to Mathematics of the Job-Shop. John Wiley and Sons, Inc. (1982).
10. Graham, R. L., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G.: Optimization and Approximation in Deterministic Sequencing and Scheduling: A survey. In Annals of Discrete Mathematics (1979).
11. Harper, F.: XML Standards and Tools. eXcelon Corporation, USA (2001).
12. Ignall, E., Schrage L.: Application of the Branch-and-Bound Technique to Some Flow-Shop Problems. In: Operations Research Vol. 13 (3) (1965).
13. Jordan, C.: Batching and Scheduling. Springer-Verlag, Germany (1996).

14. Laurent, S., et al.: Programming Web Services with XML-RPC. O'Reilly & Associates, Inc. (2001).
15. Morton, T., Pentico, D.: Heuristic Scheduling Systems. John Wiley & Sons Inc., USA (1993).
16. Pardi, W.: XML - Enabling Next-generation Web Applications. Microsoft Press, USA (1999).
17. Pinedo, M.: Scheduling Theory, Algorithms and Systems. Prentice-Hall Inc., USA (1995).
18. Ramalho, J. C., Henriques, P.: XML & XSL da Teoria à Prática. FCA, Editora de Informática Lda., Portugal (2002).
19. Varela, L., Aparício, J., Silva, S.: An XML Knowledge Base System for Scheduling Problems. In: Proceedings of the Innovative Internet Computing System Conference.: Springer-Verlag in the Lecture Notes in Computer Science series, Kuhlungsborn, Germany (2002) 61-70.
20. Varela, L., Aparício, J., Silva, S.: Scheduling Problems Modeling with XML. In: Proceedings of the 4th International Meeting for Research in Logistics. International Meeting for Research in Logistics, Inc., Lisbon, Portugal (2002) 897-909.