Combining Activity DSM with Temporal Logic for Collaborative Planning and Scheduling

Mathieu Baudin, Pierre Bonnal and Jean-Michel Ruiz

1 CERN, European Organization for Nuclear Research, Geneva, Switzerland
2 LCPI (Product Design and Innovation Laboratory) at Arts & Métiers ParisTech, Paris, France
3 École Centrale Marseille, Marseille, France

* Corresponding author E-mail: mathieu.baudin@cern.ch

Received 01 Mar 2013; Accepted 02 Jul 2013
DOI: 10.5772/56844

© 2013 Baudin et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract This work presents an approach that implements enhanced scheduling algorithms to plan and schedule interventions in scientific facilities emitting various types of ionizing radiation such as the ones present at CERN in Geneva, (Switzerland) or at GSI in Darmstadt (Germany). To deal with the collaborative process of activity creation and submission used in those organizations, we propose a framework to set the appropriate sequence or skeleton for the activities. This framework combines Allen’s Interval Algebra and DSM (Design Structure Matrix). It enables the sequence of activities to be automatically computed while gathering and taking into account the needs of users and testing their compatibility. It also deals with technical-type or resource-type constraints between activities as well as incompatible submissions. The work described in this paper includes details of the collaborative submission process and introduces suggestions for compromising as well as temporal calculations with the introduction of parameterized DSMs rather than binary ones.

Keywords Planning and Scheduling, DSM, Allen’s Interval Algebra, Radiation Safety

1. Introduction

The scientific facilities at CERN such as the Large Hadron Collider (LHC), the world’s largest particle accelerator, rank among the most complex systems ever built. To enable researchers to work at the highest energy levels ever reached in particle physics, several thousand people in a wide range of fields (e.g. cryogenics, magnet technologies, radio-frequency, beam instrumentation, radiation protection) were involved in the project. From the project approval for construction in 1994, through the completion of the R&D program to the commissioning of the beams in 2008 and the start of operations in 2010, more than fifteen years of effort were required. The technical and managerial expertise of all the contributors was a key factor in the success of this endeavour and it is still the case today for the operation, maintenance and improvement of the facilities.

Everything in particle accelerators is done to keep the need for human interventions and maintenance to a minimum, in order to avoid exposing personnel to...
activated equipment emitting ionizing radiation. However, to be able to keep (or even increase) the extremely high level of performance required by the research performed using those facilities, such interventions and maintenance are still necessary.

Several management and communication methods were developed to cope with the requirements of this large-scale project. One could for instance think of EDMS [1], which stands for Engineering Data Management System, a tool which was developed at the early development stages of the LHC project [2], or also the conference and workshop management platform INDICO [3]. The World Wide Web was also invented at CERN. What all these tools have in common is their collaborative dimension. In keeping with this collaborative culture present at CERN, the tools currently under development for intervention management also aim at enabling these to be used collaboratively and to enable the sharing of knowledge and data. This work thus focuses on techniques enabling both collaborative and optimized planning and scheduling techniques.

The rest of this article will be organized as follows: section 2 focuses first on the requirements of collaborative intervention planning and scheduling in scientific facilities emitting various types of ionizing radiation. Sections 3 develops the use of the activity DSM for planning and scheduling and section 4 describes a temporal logic framework, Allen’s Interval Algebra, used to enable collaborative constraint gathering in the form of time intervals networks. Finally, section 5 gives some insight into the computing, by describing the sequence in which the algorithms of sections 3 and 4 are used, before introducing the next steps of this research. Section 6 shortly concludes this work.

2. Intervention planning and scheduling requirements

2.1 Interventions in scientific facilities emitting ionizing radiations

Intervention management systems dedicated to the planning and scheduling of interventions such as the ones mentioned in the introduction have to be able to handle two main types of interventions:

- One of a kind, project-type interventions for the consolidation or upgrade of the facilities;
- Operations-type interventions that are of a rather repetitive nature such as corrective and preventive maintenance interventions, or inspections such as the radiation surveys.

Safety constraints have to be taken into account, especially regarding exposure to ionizing radiation which must be subject to the ALARA approach (As Low As Reasonably Achievable). ALARA is an optimization approach implemented in facilities emitting ionizing radiation in order to strike a balance between the radiation doses received by human personnel and the means (e.g. budget, available resources, etc.) that can reasonably be put into action to reduce those doses. This optimization can lead to alternative solutions to human intervention, such as the use of remotely handled intervention devices. One of the basic optimization methods for reducing received radiation doses consists of optimizing the exposure time of human or teleoperated resources, as well as their availability. This can thus be started as early as during the planning and scheduling phase of an intervention, by optimizing the sequence of tasks based on the technical and resource constraints already known to the involved stakeholders.

Finally, logistics tasks are also of importance, considering that most of the facilities are underground and some of them can only be accessed via a long section of tunnel. Logistics operations, in this context, will have to be included in the planning and scheduling phases of the interventions, because they take a non-negligible part of the intervention time and mobilize resources, tools and facilities as much as the tasks that are performed for maintenance or upgrading activities. On top of this, some transportation tasks may also have to be performed in or through radioactive areas.

2.2 Collaborative planning and scheduling

A second important aspect of our planning and scheduling problem comes from the collaborative culture of information systems at CERN and more generally in all organizations operating similar facilities.

As we already specified, only a few of these systems are centrally managed, which implies a collaborative use. In the same spirit, the framework we have developed is meant to be used by several hundreds of people who will submit several thousands of intervention requests a year. Periods of long shutdown such as the one starting at CERN in early 2013 anticipated to last until the end of 2014, are of specific interest as they will see a huge amount of interventions of all the types mentioned in the previous section.

To ensure a proper collaborative intervention submission process, any user interacting with the system should be able to specify the constraints their intervention will create or depend on, with respect to the activities already entered in the system. In such a situation, one can expect conflicts to arise in terms of activity sequence. For instance, the first user could request that a task A should be performed before a task B, while another user could assess that task A also requires information from the execution of task B and that the opposite sequence should be preferred.
One of our key challenges is to deal with conflicting collaborative submissions. Leaving the entire problem solving to a system superintendent would in fact raise two issues:

- First, the amount of work to solve manually and case by case would make this task slow and fastidious;
- Second, this would not be collaboration-compliant, since only one or a few individuals would perform most of the decision making.

We do not aim to develop a system without moderation which could easily be prone to chaos. However, we do wish to leave the preliminary phase of planning and scheduling open to suggestions from all users before implementing a stronger moderation.

To summarize, we wish to combine in our framework a large number of intervention submissions by several different users and the possibility to accept conflicting submissions. In addition, in the preliminary stages of planning and scheduling for which such a framework would be used, we do not aim to leave those conflicts to be solved by one or a few moderators. We thus concluded that it would be interesting to implement some pseudo Artificial Intelligence elements in our framework, to pre-plan and pre-schedule interventions and establish one or several possible preliminary skeletons that could later be revised by system moderators.

As for the chosen methods, this article is a development of the ideas provided in Baudin et al. [4], in which the foundations of the framework presented here were set. The general idea was to use the existing DSM method to schedule the activities and establish the optimal sequence. However, this DSM is meant to be built using data collected from several users collaborating together. To gather the needs and requirements of users, we chose to use the Artificial Intelligence framework developed by James Allen in the 1980s: Allen’s Interval Algebra. This section is dedicated to the description of the collaborative submission process and the reasons behind the choices of DSM and Allen’s algebra.

3. DSM

DSM, which stands for Design Structure Matrix, is a generic matrix-based framework introduced by Donald Steward in the early 1980s [5]. It is a powerful management tool which enables one to represent processes or systems in a clear and compact manner, with no size limitation.

As a result, it was widely applied in the 1990s in product development processes and system modelling, as developed in [6], [7], [8], [9]. In fact, it provides a high level of information regarding task dependencies (see Figure 1) and more specifically when it comes to information feedback or looping which are commonly found within engineering and design projects.

This latter characteristic is our main reason for choosing DSM. In fact, it is a handy way to represent conflicts: two activities providing data to each other can be represented as a loop, and more classic planning and scheduling methods such as PERT, CPM or PDM do not allow for the easy representation of this feedback or looping in a compact and easy way.

Other process modelling methods were also considered in a different work currently under review, such as generalized activity networks (e.g. GERT [10], [11], [12], VERT [13]) or notations such as SADT described in [14], [15] or IDEF3 (see [16]). However, here again, DSM outranks them in terms of the size (it is far more compact) of the representation. On top of this, the complexity of generalized activity networks contrasts with the simplicity of DSM’s optimization algorithms (see 3.1), while the optimization dimension is absent from SADT or IDEF3.

A DSM is a square matrix with as many lines and columns as there are activities in the modelled process. A non-zero \(a_{ij}\) matrix element (also called a mark) indicates that task \(i\) provides information to task \(j\). All marks placed on a task’s row represent its input information and the marks appearing on a task’s column represent its output information. If one reads the matrix as a sequence of activities (from top to bottom and left to right), then the marks under the diagonal stand for information transferred to later tasks while those which appear above the diagonal represent information fed back to previous tasks.

Such information feedback leads to iterations, which means that some tasks have to be performed using incomplete inputs and may need to be revised. Iterations are common for processes in many engineering fields, such as chemical engineering and product design, but also in mathematics to solve large systems of equations (see for instance [6], [9]).

3.1 DSM Analysis

Once a DSM is built, the aim of the analysis is to optimize the sequence of tasks in the matrix. For this, two stages are performed, each of them corresponding to an algorithm: first the partitioning and second, the tearing.
Those two algorithms are described in [2], [3] and [9], and detailed tutorials are available on the DSM community website [17]. We will thus only give a succinct presentation in this section.

The partitioning algorithm (also called sequencing) aims to establish an optimal permutation of the rows (or columns) of the DSM to enhance the available information in the system. The goal of this analysis step is to obtain a lower block triangular DSM in which the remaining feedback marks are concentrated along the diagonal. This is performed according to two steps:

- Step 1: Schedule independent tasks: tasks with no necessary input (empty rows) are scheduled first and tasks with no output (empty columns) are scheduled last.
- Step 2: Identify interdependent groups of tasks. The two main methods for this step are the Path Searching and the Powers of the Adjacency Matrix [6]. When a loop is identified, the tasks of which it is composed are collapsed together and considered as one task for the next steps of the partitioning.
- Step 3: Repeat steps 1 and 2 until all tasks are scheduled.

Once the DSM is partitioned, the tearing algorithm is used to re-sequence the tasks within blocks containing interdependencies and that have been collapsed during partitioning. Tearing aims at finding an ordering in those blocks to start the iterations by making assumptions regarding some elements. Two main methods are given in [5], [6]: the use of Shunt diagrams and tearing by heuristics. However, this step of DSM processing requires engineering and management judgment from the user, and involves a certain amount of subjectivity. A good knowledge of the system or of the intervention is obviously useful in choosing which activity to schedule first in the loop and to reduce its number of iterations, while a poor choice could increase it instead. A typical technique for choosing the dependencies to be “torn” is to pick the ones with the minimum number of row elements and thus with the minimum number of input streams.

Block sub-matrices can then be distinguished to form what is called a partition. These are the smallest squared matrices along the diagonal with only zeros to their right, as illustrated in Figure 2.

Figure 2. A seven task example for DSM partitioning algorithm
3.2 Building the DSM

So far, we have only described the planning and scheduling tool that DSM can be. However, we have made no mention of the collaborative dimension of our problem. We intend to take this into account in the DSM building process of our framework.

The situation we considered in section 2 involves several users collaborating on an equal level to feed their requirements in terms of constraints between tasks into the DSM. However, those constraints are not simple binary relations as described previously in this section.

They can be of different natures, for instance linked to the resource availability or the technical requirements of the facilities in which interventions are planned. In addition, since we are dealing with sequential constraints between activities, their complexity goes beyond the binary relation of input (or no input) from one task to the other. For instance, if a task A is to be performed before a task B, the users should be able to specify additional information such as a mandatory delay between A and B or, the other way round, that B has to start right after A is completed.

The combination of this complexity with the collaboration requirements led us to work on temporal logic frameworks, to express the sequential constraints.

4. Allen’s Interval Algebra

James Allen introduced the calculus that is known today as Allen’s Interval Algebra in the early 1980s [18], [19], [20]. This is a temporal logic based Boolean algebra which tackles the issue of how to represent the knowledge we have about time from an Artificial Intelligence point of view, and how to handle new pieces of information in a task network.

This issue of time representation was tackled intensely during the 1970s and 1980s, producing many different possible solutions (see for instance [21], [22], [23]). Allen’s Interval Algebra is one of these methods and it was the first to model time using an interval representation rather than a discrete model using time points such as dates or states. The general idea of this framework is to describe actions or events as the time intervals over which they take place. Each action or event also has a set of preconditions which must be true during earlier time intervals before it occurs as well as a set of effects that will be true after it terminates.

The whole algebra is based on the fact that much of mankind’s perception of time is relative. The advantage of such a representation is that one can represent events and the constraints that link them without having to worry about dates, or even fuzzy dates. This is the main reason we worked using this representation. For intervention planning in scientific facilities emitting ionizing radiation, the emphasis is placed on the fact that a task has to be performed and not on its precise duration, or starting and ending dates. In fact, engineers and technicians will actually never refer to precise durations of tasks, but use instead more imprecise terms such as fuzzy temporal descriptors (a few hours, about a week etc.). This type of description is very different from those involving starting and ending dates (which are precise time points), which are paradoxically the most commonly used nowadays. This is the case in planning and scheduling applications used at CERN or in off-the-shelf commercial software tools.

In addition, the possibility to describe the sequential constraints between tasks and to propagate the effects of new tasks in the network without having to worry about dates and duration is a critical advantage of such representation. These processes are explained in the following paragraphs.

Allen’s Interval Algebra allows for a precise description of relations between time intervals. In fact, Allen defines the relations in [18] thirteen of them (actually seven relations and their inverse relations) as shown in Figure 3. Using a composition or transitivity table, it is possible to build transitivity relations and point out incoherent elements in a set of time intervals.

![Figure 3. Conversion from Allen’s interval algebra to DSM](www.intechopen.com)
assembly or production lines. For instance, Smith et al. describe a possible use of interval based planning and scheduling algorithm for an “ambitious spacecraft” [25]. The advantage of planning and scheduling systems based on time-interval representations is that they manage to reduce the gap between planning and scheduling, which are often considered as two distinct problems in Artificial Intelligence: planning problems tend to deal with action selection and cascading levels of interaction between them while scheduling problems are more concerned with setting an optimized sequence of already selected tasks given their constraints and resource needs.

Time-interval-based logic has also been occasionally used in academic fields other than artificial intelligence. For instance, some publications in project management can be mentioned [26]. Moreover, references [27], and [28] describe workflow modelling and business process execution using interval based constraint networks, which is even more relevant for the computation framework presented in this article.

To conclude this presentation section, it can be added that extensions to the basic definition of the interval time representation have been introduced. In [18], Allen already introduced reference intervals which aims to reduce computation time for constraint propagation algorithms by creating clusters of interrelated time intervals. For instance, in our case, an intervention can be considered to be the reference interval of all the tasks which compose it. The bases for duration calculation algebra based on a similar constraint propagation algorithm, are also introduced.

It was also shown in [20] that the thirteen time relations can all be further decomposed and expressed using only one of them, the MEET relation (see Figure 3), as a primitive. These possibilities of combination or decomposition can be used in our case to introduce constraints which do not appear in the original algebra: for instance, by combining BEFORE, AFTER, MEETS and IS MET, one can indicate that two tasks have to be separated, regardless of which one is to be performed first. Additionally, in [20], the notion of moment as a non-decomposable time interval was introduced to deal with the time-point notion present in discrete-time models, and the definition of starting and ending points of an interval was also formalized. This notion of time-point to extend the interval algebra was also studied by Zaidi in temporal logic programming at the end of the 1990s [29], [30] and can be useful to include dates at the end of the planning process.

4.1. Constraint propagation and transitive closure

The main role of the algebra itself is to encode the relations between time intervals. The job of reasoners is to compute the relations entered as input, check whether they form a compatible set and compute what can be inferred from them. This process is referred to as “determining the transitive closure” of a set of assertions.

```
While (Queue not empty):
    get next pair <i, j> in Queue:
    for each k ≠ j Comparable to i:
        Temp = Constraints(<i, j>, <i, k>, <j, k>)
        if Temp = Ø
            then signal contradiction
        else if Temp ≠ <i, k>
            then <i, k> >= Temp
            append <i, k> to Queue
    end
    for each k ≠ i Comparable to j:
        Temp = Constraints(<j, i>, <i, j>)
        if Temp = Ø
            then signal contradiction
        else if Temp ≠ <k, j>
            then <k, j> ≥ Temp
            append <k, j> to Queue
    end
```

Figure 4. Constraint propagation algorithm

This task is performed using a constraint propagation algorithm which is given in Figure 4. Every time the matrix containing temporal constraints between tasks is updated, the modified matrix elements <i, j> are added to a first-in-first-out queue. Then, for every pair <i, j> removed from the queue, the algorithm checks if constraints between other pairs of tasks <i, k> or <j, k> are affected. If it is the case, the related pair is added to the queue. If the new constraint for element <i, j> is not compliant with the transitivity table, backtracking is performed.

In [18], Allen suggests that this algorithm will run to completion in polynomial time and that $O(n^2)$ calls are necessary for the procedure. Vilain and Kautz however show in [31] that $O(n^3)$ operations are needed for the algorithm to run to completion.

However, even this polynomial number of operations is only true for three-node sub-networks and Allen hints that completing the full closure (checking coherency and then inferring new relations) of a time interval network may be in fact exponential. This is confirmed and proved in [31]: completing closure is an NP-hard problem, which means that it cannot be solved in polynomial time and that the algorithm behaves exponentially.

In fact, while the algorithm will never make an invalid inference about constraints between three intervals, it does not deal with larger sets of intervals, which would be required to calculate full closure. An example of an inconsistent network is given by Allen in [18] and reproduced in Figure 5. For any group of three nodes, the transitivity relations can be verified, since at least one of
the two relations carried by the arcs (except between B and D where there is only one) will fit in the transitivity table. However, one can show (for instance by imposing \( A f C \) instead of \( A (f, fi) C \) that it is impossible to get one and only one symbol per arc considering the four nodes together.

Several strategies can be adopted to deal with this intractability:

- Limit ourselves to small databases of about a dozen intervals, which is reasonable for describing interventions;
- Use one of the eighteen tractable subalgebras of Allen’s Interval Algebra [32];
- Use exponential algorithms as per Malik and Binford [33];
- Keep the polynomial-time constraint propagation algorithm however incomplete it may be. This method is actually efficient for systems such as ours, in which we mainly gather the time relations and do not make many inferences (see section 5.1.). In fact, we do not aim to compute new relations, but only to check that a new constraint added in the network by a collaborator fits with the previously submitted ones. This step of the algorithm is feasible in polynomial time and is sufficient in our case.

We are thus going to work within the first and fourth strategies.

### 4.2. Constraint gathering for intervention planning

The use of Allen’s interval algebra to gather user’s needs and requirements for temporal constraints grants them an expressivity larger than the binary DSM while providing a formal background to their submissions.

Our aim is, as a consequence, to have users submit tasks in the system as well as their temporal constraints to already entered tasks when such constraints exist and then, once the closure is computed, a simple conversion to DSM is easy to perform based on the correspondence between DSM and the temporal relations given in Figure 3.

To deal with the intractability problem of the algebra, we are also working with one more restriction and studying the possibility of accepting only one constraint between two tasks. This would in fact prevent users from creating problems with no overall marking.

However, this solution has some drawbacks. For instance, Allen’s interval algebra cannot state that two intervals are disjoint. For this reason, a combination of four relations is needed (BEFORE or AFTER or MEETS or IS MET).

### 5. Computation

#### 5.1 Sequence of algorithms

The proposed algorithm, or rather sequence of algorithms, is performed every time a collaborator submits an activity that may have an effect on already existing activities. After the user has added a new activity, it basically consists of 5 steps:

- Step 1: add a row and a column to the activity DSM;
- Step 2: implement the constraints between activities (new and previously entered) in the DSM with the Allen interval algebra notation thus building a parameterized DSM;
- Step 3: check the compatibility of all constraints using the transitivity rules of Allen’s interval algebra. If all assertions are compatible, step four can be performed. Otherwise, a compromise algorithm has to be run (see section 5.2.);
- Step 4: convert the parameterized DSM with Allen’s Interval Algebra notations into a classical binary DSM. Other possible parameterized DSMs are described in section 5.2.;
- Step 5: run a classic DSM partitioning algorithm to establish the sequence of activities and perform a tearing algorithm if loops are identified.

**Figure 6** illustrates the collaborative elaboration of the sequence of activities necessary to change an instrument on a beam line. Three groups are involved, and they submit their activities at \( t_1 \), \( t_2 \) and \( t_3 \). Seven tasks in total are to be scheduled. For the two first activity submissions at \( t_1 \) and \( t_2 \), the five steps described previously are performed with no trouble. The submission at \( t_3 \) results in a conflict and a compromise is needed.
Intervention: changing a beam instrument

| Task | Modus operandi | Resource |
|------|----------------|----------|
| 1a   | disconnect instrument signal cables | E        |
| 1b   | disconnect controls signal cables  | E        |
| 2    | replace instruments               | I        |
| 3    | replace controls devices          | C        |
| 4a   | reconnect instrument signal cables | E        |
| 4b   | reconnect controls signal cables  | E        |
| 5    | test the system                   | E + C    |

Figure 6. Illustration of the collaborative process of intervention submission. Three groups (E, I and C) collaborate to plan and schedule the necessary activities for replacing a beam instrument.

In fact, the group in charge of controls submits that the control devices will be changed (task 3) after the beam instrument (task 2), which would be noted: 2 b 3. However, they also submit that operation tests (task 5) will be performed right away, which gives 3 m 5, to allow personnel to perform the two tasks in a row instead of having to exit the facility before coming back for the tests. This last assertion creates a conflict: in fact, the group in charge of electrical cabling (group E) needs to reconnect the control cables (task 4b) between tasks 3 and 5 as indicated in their submission at t3.

So to summarize, we have so far a framework that allows users to:
- Submit activities and indicate their dependencies in terms of temporal relations;
- Check whether the constraints they specify fit with the activities and constraint sets previously entered into the system;
- Establish an optimized skeleton of activities using a conversion from Allen’s Interval Algebra to DSM followed by a partitioning algorithm.

We already have some ideas as to how to compromise between conflicting submissions and this issue will now be discussed. Furthermore, since our goal is to establish a schedule for the activities, we will also introduce some insight into how we expect to add duration reasoning with parameterized DSMs.

5.2 Further developments

To perform a compromise in a case of conflicting submissions, we have explored two main possible solutions.

The first and easiest one to implement involves considering activities for which conflicting constraints exist as being coupled and so they should be considered as a loop while computing the DSM. A tearing algorithm can be performed to optimize their sequence inside of the loop, but a rework of one or more of them may be needed.

The second solution is that which appears in Figure 6 as illustrated with the different types of arrows. It involves discriminating between “resource constraints” and “technical constraints” to establish which one should have
the priority and then trying to “relax” the resource constraints if possible (i.e. replacing them with a weaker constraint, such as by changing “m” to “b” to allow for a delay between two tasks). If we take the example of Figure 6, it can be easily deduced that the operations tests in task 5 cannot be performed if all the cables are not reconnected (tasks 4a and 4b). This is a technical constraint and the resource constraint “m” imposed by group C between tasks 3 and 5 cannot be prioritized over it.

However, for such a compromise to be successful at least in theory, the resource constraint obtained after being “relaxed” needs to fit with the transitivity relations of Allen’s Interval Algebra. Additionally, an ordering of the degree of constraint imposed by each of the thirteen relations needs to be defined. Although it is clear that a MEETS relation will be a stronger constraint than a simple BEFORE, it is not that clear when it comes to DURING and OVERLAPS relations.

Further work will thus focus on how to rank the time relations in such a way that a “relaxation” will still fit in the transitivity table.

It should also be noted that a binary DSM does not carry as much information as the time interval relations that are submitted in the first place. A Gantt-type diagram can still be generated after partitioning and tearing, provided that the durations of the activities are known. However, some types of parameterized DSM could encode more information, such as the delay between the start of two overlapping tasks or between two tasks constrained by the BEFORE relation.

Instead of marks, one could for instance avoid such information losses using an index linked to duration. The DSM mark would take a numerical value lower than one in case of an overlap and higher than one in case of a delay after completion of the preceding task. A value of one would then correspond to the MEETS relation of Allen’s Interval Algebra.

Finally, the situation described above might actually never be encountered in engineering projects, where precise dates and durations are often missing, and where one often has to deal with imprecise temporal knowledge. In such cases, using fuzzy durations could be tempting, as suggested in [34] and shown in Figure 7. A duration can be associated to a fuzzy interval, thus enabling several possible durations (and as many possible schedules) for the same activity based on one single linguistic descriptor (a small week, about an hour etc.).

Figure 7. Example of linguistic descriptors to characterize duration. Figure extracted from [32].

6. Conclusions

We propose a framework for intervention planning and scheduling which enables users working in collaborations to submit intervention requests and to compute the optimal sequence of activities. The interest of such a framework for scientific facilities emitting ionizing radiation lies in the optimization of exposure time both for human personnel and tools for teleoperated intervention, as early as the planning and scheduling phase of an intervention. The methods chosen for our framework, and especially Allen’s Interval Algebra, enable potential users to submit activities and their interdependencies with a wide range of relationships. This allows for a rich expressivity, while the formalism behind the transitivity table and DSM enables us to deal with incompatible submissions and then to establish an optimized sequence of tasks. Further works will focus on the issues linked to compromising in cases of conflicting submissions and the better encoding of temporal information in parameterized DSMs.

7. Acknowledgments

This research project has been supported by a Marie Curie Actions Initial Training Network Fellowship of the European Community’s Seventh Framework Programme (FP7) under contract number (PITN-GA-2010-264336-PURESAFE).

8. References

[1] “CERN EDMS Service Website,” 22-Apr-2012. [Online]. Available: http://espace.cern.ch/edms-services/services/default.aspx.

[2] G. Bachy and A.-P. Hameri, “What to be implemented at the early stage of a large-scale project,” International Journal of Project Management, vol. 15, no. 4, pp. 211–218, Aug. 1997.

[3] “CERN’s INDICO website.” [Online]. Available: http://indico.cern.ch.

[4] M. Baudin, P. Bonnal, and J. De Jonghe, “A DSM Based ’2.0’ System for Human Intervention Planning and Scheduling in Facilities Emitting Ionizing Radiations,” Proceedings of the 14th International Dependency and Structure Modelling Conference, Kyoto, Japan, 2012.

[5] D. V. Steward, “The Design Structure System: A Method for Managing the Design of Complex Systems,” IEEE transactions on Engineering Management, vol. 28, no. 3, 1981.

[6] D. A. Gebala and S. D. Eppinger, “Methods for Analyzing Design Procedures,” Proceedings of 3rd International ASME Conference on Design Theory and Methodology, Miami, Florida, 1991.
A. A. B. Pritsker and W. W. Happ, “GERT: Graphical Evaluation and Review Technique. Part I. Fundamentals,” The journal of Industrial Engineering, vol. 17, pp. 267–274, 1966.

A. A. B. Pritsker and G. E. Whitehouse, “GERT: Graphical Evaluation and Review Technique. Part II. Probabilistic and Industrial Engineering Applications,” The journal of Industrial Engineering, vol. 17, pp. 293–301, 1966.

G. E. Whitehouse and A. A. B. Pritsker, “GERT: Part III - Further Statistical Results; Counters, Renewal Times and Correlations,” AIEE Transactions, vol. 1, pp. 45–50, 1969.

G. L. Moeller and L. A. Digman, “Operations Planning with VERT,” Operations Research, vol. 29, pp. 676–697, 1981.

D. T. Ross and K. E. S. JR, “Structured Analysis for Requirements Definition,” IEEE Transactions on Software Engineering, vol. SE-3, no. 1, pp. 6–15, 1977.

D. T. Ross, “Structured Analysis (SA): A Language for Communicating Ideas,” IEEE Transactions on Software Engineering, vol. SE-3, no. 1, pp. 16–34, 1977.

“IDEF: Integrated DEFinition Methods.” [Online]. Available: http://www.idef.com.

U. Lindemann, “The Design Structure Matrix (DSM),” 2009. [Online]. Available: http://DSMweb.org.

J. F. Allen, “Maintaining Knowledge about Temporal Intervals,” Communications of the ACM, vol. 26, no. 11, pp. 832–843, 1983.

J. F. Allen, “Towards a general theory of action and time,” Artificial Intelligence, vol. 23, no. 2, pp. 123–154, Jul. 1984.

J. F. Allen and P. J. Hayes, “A Common-Sense Theory of Time,” Proceedings of the International Joint Conference on Artificial Intelligence, Los Angeles, CA, 1985.

B. C. Bruce, “A model for temporal references and its application in a question answering program,” Artificial Intelligence, vol. 3, pp. 1–25, 1972.

D. McDermott, “A temporal logic for reasoning about processes and plans,” Cognitive Science, vol. 6, no. 2, pp. 101–155, Apr. 1982.

K. Kahn and G. A. Gorry, “Mechanizing temporal knowledge,” Artificial Intelligence, vol. 9, no. 1, pp. 87–108, Aug. 1977.

J. F. Allen, “Planning as Temporal Reasoning,” Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, 1991.

D. E. Smith, J. Frank, and A. K. Jonsson, “Bridging the Gap Between Planning and Scheduling,” Knowledge Engineering Review, vol. 15, no. 1, 2000.

S. Hussain, “Project Management Using Temporal Logic Programming,” 2000 IEEE Int. Conference on Systems, Man, and Cybernetics. Nashville, TN, pp. 2144–2149, 2000.

Ruopeng Lu, Shazia Sadiq, Vineet Padmanabhan, and Guido Governatori, “Using a Temporal Constraint Network for Business Process Execution,” Proceedings of the 17th Australian Database Conference, Hobart, Australia, pp. 157–166, 2006.

V. Dufour-Lussier, F. L. Ber, and J. Lieber, “Extension du formalisme des flux opérationnels par une algébres temporelle,” arXiv:1209.5664, Sep. 2012.

A. K. Zaidi, “On temporal logic programming using Petri nets,” IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 29, no. 3, pp. 245–254, May 1999.

A. K. Zaidi and A. H. Levis, “TEMPER: a temporal programmer for time-sensitive control of discrete event systems,” IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 31, no. 6, pp. 485–496, Nov. 2001.

M. Vilain and H. Kautz, “Constraint Propagation Algorithms for Temporal Reasoning,” Proceedings of the Fifth National Conference on Artificial Intelligence, Philadelphia, PA, 1986.

A. Krokhin, P. Jeavons, and P. Jonsson, “Reasoning about temporal relations: The tractable subalgebras of Allen’s interval algebra,” J. ACM, vol. 50, no. 5, pp. 591–640, Sep. 2003.

J. Malik and T. O. Binford, “Reasoning in Time and Space,” Proceedings of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, pp. 343–345, 1983.

P. Bonnal and M. Baudin, “Handling a Design Structure Matrix Based on Fuzzy Data,” Proceedings of the 14th International Dependency and Structure Modelling Conference, Kyoto, Japan, 2012.