# Modelling SBR cycle management and optimization using Events and Workflows

D. Sottara

P. Mello

G. Morlini

F. Malaguti

L. Luccarini

*International Environmental Modelling and Software Society (iEMSs)*
*2012 International Congress on Environmental Modelling and Software*
*Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany*
*R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)*
*http://www.iemss.org/society/index.php/iemss-2012-proceedings*

# Modelling SBR cycle management and optimization using Events and Workflows

**D. Sottara** [a], **P. Mello**[a], **G. Morlini**[a], **F. Malaguti**[a] **and L. Luccarini**[b]

[a]*DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna (BO), Italy (davide.sottara2@unibo.it, paola.mello@unibo.it, gabriele.morlini@unibo.it, filippo.malaguti@unibo.it*

[b]*ENEA – UTVALAMB–IDR, Via Martiri di Monte Sole 4, 40129 Bologna (BO), Italy (luca.luccarini@enea.it)*

**Abstract:** Sequencing Batch Reactors (SBRs) provide several advantages in terms of flexibility and robustness to variations in the inflow concentrations and sludge alterations. Adjusting the duration of the load, reaction and discharge phases, it allows to optimize the treatment, saving time and energy. Optimal policies can be defined by observing and analyzing some chemical and physical parameters such as pH, redox potential and dissolved oxygen concentration. Various Artificial Intelligence (AI) techniques have been proposed and used to recognize the state of the biological processes inside the plant, using the signal trends and changes as indirect indicators. In particular, the termination of a process (typically, denitrification in the anoxic phase or nitrification in the aerobic one) can be estimated by a management system and used in control policies. In this paper, we point out that this recognition task is only part of the responsibilities of a potential Environmental Decision Support System (EDSS) managing an SBR, and by no means the only one which can take advantage of AI techniques. In fact, the entire control and management system could be defined and implemented using declarative AI techniques, deployed within a uniform execution environment. In particular, we propose two alternative but similar models of the SBR operation cycles: one is based on workflows, exploiting the BPMN2 (Business Process Management Notation v.2) standard for the definition and execution of business processes; the other instead is founded on the principles of (Reactive) Event Calculus. Both representation capture the operational behavior of the SBR, supporting the state transitions and the actions associated to each state. The transitions themselves are driven by events, i.e. relevant state changes. The events are identified either directly, through the sensor system installed on the plant, or analysing a combination of other more elementary events, and conditioned by the actual state of the plant. The correlations between events, states and control actions, as well as their consequences, will eventually be defined using rules, which will also encode the necessary knowledge to deal with exceptional conditions, accounting for a more flexible system.

***Keywords:*** Sequencing Batch Reactors, Business Process, Event Calculus

## 1 Introduction and Related Works

Sequencing Batch Reactors SBR are a particular class of activated sludge waste-water treatment plants (WWTPs), with a layout characterized by the presence of a single treatment tank where all the necessary reactions take place sequentially in time. SBRs have several applications, but when used for urban waste-water treatment, the main reactions involved are the *nitrification* and *denitrification* of nitrogen compounds and the simultaneous removal of organic matter.

Due to the necessity of different conditions in the tank, a SBR operates in a cyclic modality: a batch of water is loaded, treated alternating anoxic and aerobic conditions, and finally discharged. The effectiveness of the treatment process depends on many factors, including the hydraulic retention time (HRT): roughly speaking, this factor determines how long the polluted water remains in a tank and thus the amount of time it is subject to the particular environmental conditions which favour the different bio-chemical reactions. Thus, the timing of the phases must be chosen carefully, finding a trade-off between ensuring that the HRT is sufficient to complete the reactions and the contingent necessity to process a certain number of batches every day, implicitly determining a limit to the maximum duration of a cycle.

Unfortunately, the effective load of pollutants in the influent is not constant, but varies with factors such as the time of the day (e.g. few pollutants are discharged in the sewers at night, when people sleep), the period of the year (e.g. the population of an area may vary significantly during holidays), the presence of periods of intense rainfalls or draught, etc .... In practice, the maximum duration of the phases is often tuned on the average case, plus a relevant safety margin, to ensure that the process can be successful. However, this conservative approach may lead to a waste of time (the reactions are usually completed well before the deadlines), energy (air blowers are expensive) and, ultimately, opportunities (with shorter cycles, more water can be treated).

For this reason, a relevant amount of studies have been performed on how to observe the process' progress in real time, often using cheap sensors such as pH, redox potential (ORP), dissolved oxygen concentration (DO) and temperature (T), so to trigger phase changes as really needed, rather than as estimated a priori. Various statistics or artificial intelligence techniques have been applied, including neural networks (Fan and Xu [2007]), PCA (Ciappelloni et al. [2006]) and fuzzy logic (Traore et al. [2005], Marsili-Libelli [2006]).

The optimal management of an SBR, however, is not limited to the optimization of the process phase duration, which is a special case of control policy applied after a diagnostic check of the process status (assuming it results in a diagnosis of "good health"). Like other classes of WWTPs, SBRs can be monitored and controlled by Knowledge-Based Environmental Decision Support Systems (Cortes et al. [2001]) in order to cope with various types of situations, ranging from routine management to dealing with specific situations such as malfunctionings, anomalous intakes or exceptional operating conditions imposed by the plant manager.

One of the advantages of EDSS is their modularity: given an appropriate infrastructure (Denzer [2005]), the knowledge base used by the system can usually be customized, deploying and integrating any module implementing a required or desirable feature. Modules can be general-purpose, providing common services (e.g. data storage or visualization, statistical analysis), or task-specific (e.g. optimizing the nitrogen removal in a tank) and are typically implemented using service- (Sottara et al. [2009]) or agent- oriented architectures (e.g. see Ceccaroni [2001]).

In this paper, we address the creation of a knowledge-based module specific for SBRs. The goal of this module is to manage the current state of the plant, making sure that the transitions between one phase and another take place under the supervision of the EDSS. Ideally, the EDSS should be able to replicate the plant manager's decisions, supported by a combination of experience, literature recommendations and contingent constraints (e.g. legal or environmental). Moreover, the system should be flexible enough to accomodate for changes or additions to the criteria. In fact, detecting the completion of a process (chemical reactions as well as loads/discharges) might not be sufficient to determine that the plant can safely switch from one phase to another. In real systems, the optimization of the treatment process is usually only one of the goals, which may also include cost minimization or managing anomalies. So, on a case by case basis, it might be necessary to check that other preconditions are met: one might want, for example, to extend the duration of a phase to reach a minimum rather than a maximum duration, or exploit an extended phase to reequilibrate a chemical parameter such as pH. Even then, when the system has determined that it is possible to switch to a given phase, decisions have to be made on which actuators (pumps, blowers, etc ...) to activate in order to enforce the necessary reaction conditions. Finally, the EDSS should monitor that the desired effect (i.e. the phase switch) has really taken place as an effect of the actuations.

To this end, the EDSS needs to be in charge of the phase management, having an explicit knowledge of what the current phase is, whether and when a switch can occur, what should be done to make the switch happen and what to expect to make sure that the switch has taken place correctly. Moreover, these concepts should be kept separate and sufficiently decoupled, in order to adapt or customize each part without affecting the others.

For the scope of this paper, then, we focus on the state management aspects. We assume that the detection and the validation of the phase switch commands has been implemented using any appropriate technique, ranging from manual commands to a complex AI. The detection subsystem is only required to comply with an "interface", i.e. generate events (timed notifications with a data payload), namely `Next` to switch to the next phase and `Switch(X)` to jump to a specific phase `X`. Regardless of the particular strategy used to generate these events (e.g. see Luccarini et al. [2010] or Marsili-Libelli [2006]), the proposed state management module will consume them in order to control the plant. Internally, we propose the use of two complementary modelling strategies, with slightly different properties, which could be used independently or even combined, according to the manager's requirements.

The SBR operation cycle has beed modelled using state machines or Petri nets (Bortone G. [2002]), but here we argue that it can also be abstracted and formalized in a more flexible and declarative way using the Event Calculus (Kowalski and Sergot [1986]). As a counterpart, its concrete execution can be conveniently modelled as a business process (BP) and thus formalized using the standard BP modelling techniques and languages (BPMN2). The two approaches, as will be discussed in the following sections, are implemented using, respectively, rules and workflows, so that they can be deployed directly in an execution environment, the business knowledge integration platform Drools[1].

**Pilot Plant** The system we will discuss in this paper has been designed to be part of an Environmental Decision Support System for the a pilot-scale SBR plant, built by ENEA and installed at the municipal plant of Trebbo di Reno (Bologna, Italy). The plant, with a working volume of $500L$ ($150L$ of working load), is fed with real sewage drawn after grit removal, has an overall hydraulic retention time of $20$ hours and sludge retention time of $15 - 20$ days. It is equipped with a set of actuators (a mechanical mixer, a variable-flow blower connected to a membrane diffuser and three pumps for influent, effluent and sludge loading and discharge) and a data collection system sampling pH, ORP and DO signals with a rate of $1$ sample/min. Its default operating cycles consist in 7 phases: during each phase, and unless specified otherwise, the actuators are activated according to the default configuration given in table 1.

[1] http://jboss.org/drools

| Phase | Feed Pump | Draw Pump | Waste Pump | Mixer | Blower |
|---|---|---|---|---|---|
| Fill | X | | | X | |
| Anoxic | | | | X | |
| Aerobic | | | | X | X |
| Waste | | | X | X | X |
| Settling | | | | | |
| Draw | | X | | | |
| Idle | | | | | |

Table 1: Defautl Actuator Configuration by Phase

Table 2: The Event Calculus ontology.

| | |
|---|---|
| $happens\_at(Ev, T)$ | Event $Ev$ happens at time $T$ |
| $holds\_at(F, T)$ | Fluent $F$ holds at time $T$ |
| $mvi(F, [T_1, T_2])$ | $(T_1, T_2]$ is a maximal validity interval for $F$ |
| $initially(F)$ | Fluent $F$ holds from the initial time |
| $initiates(Ev, F, T)$ | Event $Ev$ initiates fluent $F$ at time $T$ |
| $terminates(Ev, F, T)$ | Event $Ev$ terminates fluent $F$ at time $T$ |

## 2 An Event Calculus Model of the SBR operation cycles

### 2.1 Event Calculus

The *Event Calculus* (*EC*) is a well known formalism first proposed in 1986 by Kowalski and Sergot (Kowalski and Sergot [1986]). It was introduced to overcome the limitations (namely the frame problem) of the Situation Calculus (ST) (McCarthy and Hayes [1969]), a former similar formalism. Both are used to represent the effects of actions and changes on a domain, but EC focuses on *events* (significant point-in-time changes observed in a domain), while ST focuses on transitions and actions.

The base formulation of the EC consists of a small set of simple logical axioms that correlate the happening of events at certain times to domain's properties – typically termed *fluents* – and the time intervals at which they hold. At any time, then, the fluents define the overall state of the systems through its relevant properties. Table 2 contains the whole list of EC's axioms together with definition.

During the years, many extensions of the EC have been proposed. The *Cached Event Calculus* (*CEC*, Chittaro and Montanari [1994]) is one of those extensions; it was theorised to address an efficiency issue of the standard EC. The EC, in fact, requires to recalculate anew the current state of the domain each time a new event is acknowledged; the CEC, instead, allows the incremental update of the fluents. The variation we adopt is a transposition of the *Reactive Event Calculus* (*REC*, Chesani et al. [2010]): a lightweight form of CEC whose better performances are due to some optimisations based on the assumptions that the events are usually processed in the same order in which they happen. In particular, we have created an implementation of REC based on *Drools* and its temporal reasoning and complex event processing module, *Drools Fusion*.

## 2.2 SBR EC Model

In order to apply EC to an SBR, we simply have defined a fluent for each SBR phase and queries to check the state of the fluents. So, the state of `IdlePhase`, `LoadPhase`, `AnoxicPhase`, `AerobicPhase`, `SettlingPhase`, `DrawPhase` and `DischargePhase` determines whether the current cycle is in a given phase or not. Initially, the idle phase holds: later, the events `Next` and `Switch` (in turn generated by various other events) trigger the clipping and/or declipping of the appropriate phase fluents. More specifically, fluents always mirror the plant state at the current time, but each fluent is associated to one or more *mvi*s to keep track of the history, i.e. the time intervals when a phase was active.

A snippet of the rule base is presented in listing 1.

Listing 1: EC Model of the SBR States (part)

```
declare Next extends Event
    @role(event)
end
declare Switch extends Event
    @role(event)
    phase : String
end

declare Phase extends Fluent
    name : String
end
declare IdlePhase extends PhaseFluent end
declare AnoxPhase extends PhaseFluent end

query idle( IdlePhase $f )
        $f := IdlePhase()
end

query idlePhaseHolds()
        IdlePhase( state == true )
end

query current( Phase $f )
        $f := Phase( state == true )
end


rule "JumpToIdle"
    when
        $event: Switch( $to: phase == "Idle", $time: start )
        not ?idlePhaseHolds()
        ?idle( $next; )
        ?current( $curr; )
    then
        retract( $event );
        insert(new Clip( $time, $event, $curr ));
        insert(new Declip( $time, $event, $curr ));
end
```

The code uses a combination of forward (rules) and backward chaining (queries). The events are conditioned by the state of fluents: if the appropriate context conditions hold, the events will alter the state of the system by changing the state of the fluents. The fluents can be queried at any time to see their current state, but will also create MVIs to support queries for past times. The state fluents effectively memorize the current state, so they appear in the premise of rules managing the plant. One of such rules is used to (de)activate the actuators, while others are optionally used to monitor the plant and see that the actuators are currently responding to the commands. Likewise, they can be used within the (next) phase detection subsystem, effectively closing the loop.

For the system to work, we assume that `Switch` and/or `Next` are generated or retracted appropriately. In case the detection modules fail, we still enforce a maximum phase duration using simple rules like the one in listing 2:

Listing 2: Timeout management

```
rule "Timeout"
    when
        $x : AnoxPhase( $current : name, state == true )
                not ( Switch( this after[0, 90m] $x )
                      or Next( this after[0, 90m] $x ) )
    then
        insert( new Next() );
end
```

## 3 A Workflow Model of the SBR operation cycles

When seeing the problem from the plant's (manager) perspective, the water treatment cycle can be considered a business process: it is a sequence of structured and correlated activities which, together, contribute to the achievement of a final goal. The BPMN2[2] standard notation provides the concepts necessary to describe the process states: in particular, tasks (modelling activities), events (modelling signals), timers (required to keep track of durations) and join/fork control points (modelling branches in the process).

A SBR treatment cycle can then be modelled as a cyclic business process and its typical phases (idle, load, anoxic/anaerobic, aerobic, settling, draw, discharge) define the tasks in the workflow which must be executed to complete the process.

The completion and the subsquent flow from one task to the following is triggered by events: either an explicit `Next`, or a timeout event triggered by a watchdog timer. The basic cyclic workflow requires some additional logic to support `Switch` events since. Such a command breaks the cyclic nature of the process, so we effectively terminate the current process, executing any necessary action to put the plant into a safe state, and restart the process from the chosen phase. In particular, the model shown in figure 1, presents the cyclic workflow representing the process. In addition to the branching logic to control the process flow, the various task nodes can be divided in two classes: the plant tasks, corresponding to the treatment phases, and the control tasks, modelling the actuations required to configure the plant before and after a phase. The former is a variation of the concept of BPMN2 human task and its execution is not directly under control of the BPMN2 engine, which waits for a `Next` event (or timeout) signalling its completion or abortion; the latter, instead, are directly executed by the engine, interfacing the actuators or other infrastructure components (e.g. logs, GUI, etc. . . ).

The BPMN2 specification is directly executable in Drools, using its jBPMN twin module, and offers practical advantages such as logging and a monitoring console which can be used, at least in prototypical applications. Moreover, the business process is observable, since it generates events whenever a task or transition takes place, so it can be easily integrated with other systems. Finally, it supports the concept of user and allows to keep track of who is scheduling or commanding which action.

### 3.1 Combining the Models

The two proposed models can easily be combined and integrated. The Event Calculus is more suitable as a monitoring tool, defining and recognizing the transitions between the process states. The Business Process notation, instead, is more operational and suitable to define which actions should be executed – and when – during a process.

In a combined deployment, the `Next` or `Switch` events, clip and declip the Phase Flu-

[2] http://www.omg.org/spec/BPMN/2.0/

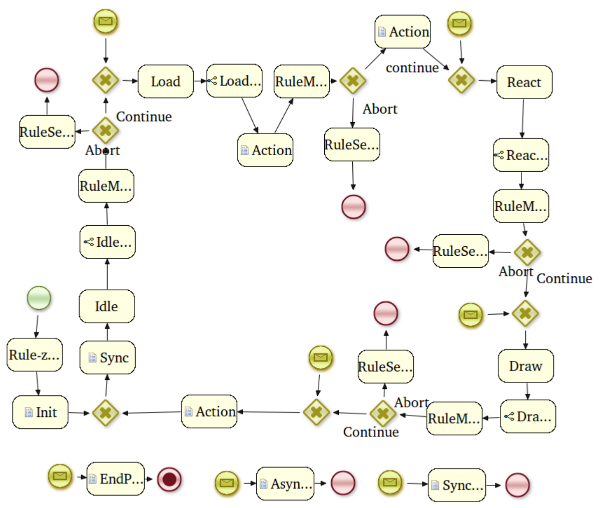

Figure 1: SBR Cycle as a Business Process

ents, provided that any context condition allows that. The eventual (de)clip of the phases determines the start or the completion of the phase tasks, which are effectively synchronized with the phase fluents. In this setting, all the procedural actions required to configure the plant are moved from the rules to the action tasks in the workflow, which are more suitable to model them. The phase deadline management is divided between the two models: the rules encode "logical" timeouts, which are based on explicit policies defined by the operators. The timers in the workflow, instead, are set with even later deadlines: the rules, then, still contain the declarative knowledge about the maximum phase duration, while the workflow timers can effectively act as failover watchdogs.

## 4 Conclusions

This paper proposes two alternative but complementary formal definitions of a SBR operational cycle. In particular, we claim that the same (semi-)declarative techniques normally used for process diagnosis and optimization can also be used to model the more procedural parts of the application, posing the basis for the development of a fully knowledge-oriented management system. We have modelled the SBR processes and phases as fluents, holding when a process or phase is actually taking place, and defined the basic events to commute from one phase, which is terminated, to the next one, which is initiated. These events, in turn, are generated as a consequence of various external signals, from manual commands to DSS-driven recommendations based on a real-time analysis of the plant sensor data. We have also shown that SBR cycles can be modelled as cyclic business processes, alternating plant tasks with control tasks: the former correspond to the proper treatment phases and the latter correspond to the control and configuration actions. We have argued that the two approaches are complementary, since the former is more suitable to monitor the state of the plant and the latter is more appropriate to define the procedural and operational aspects. Nevertheless, both formal models can be directly deployed in an execution environment and applied to a plant, either as a simple, standalone control system or as part of a larger EDSS.

## Acknowledgments

We wish to thank HERA s.r.l. for providing financial and logistic support for this work, which was also partially funded by the Department of Computer Science, Electronics and Systems "DEPICT" project.

## References

Bortone G., Longhi S., L. L. R. P. P. E. On-line control of a SBR reactor for the biological wastewater treatment. In *Proceedings the 9th Mediterranean Conference on Control and automation, Dubrovnik, June 27-29*, pages 58–65, 2002.

Ceccaroni, L. What if a wastewater treatment plant were a town of agents, 2001.

Chesani, F., P. Mello, M. Montali, and P. Torroni. A logic-based, reactive calculus of events. *Fundamenta Informaticae*, 105(1):135–161, 2010.

Chittaro, L. and A. Montanari. Efficient handling of context-dependency in the cached event calculus. In *Proc. of TIME'94 - International Workshop on Temporal Representation and Reasoning*, pages 103–112, 1994.

Ciappelloni, F., D. Mazouni, J. Harmand, and L. Lardon. On-line supervision and control of an aerobic SBR process. *Water science and technology : a journal of the International Association on Water Pollution Research*, 53(1):169–77, 2006.

Cortes, U., M. Sanchez-Marre, R. Sangueesa, J. Comas, I. R.-Roda, M. Poch, and D. Riano. Knowledge management in environmental decision support systems. *AI Communications*, 14(1):3–12, 2001.

Denzer, R. Generic integration of environmental decision support systems-state-of-the-art. *Environmental Modelling & Software*, 20(10):1217–1223, 2005.

Fan, L. and Y. Xu. A pca-combined neural network software sensor for sbr processes. In *ISNN '07: Proceedings of the 4th international symposium on Neural Networks*, pages 1042–1047, Berlin, Heidelberg, 2007. Springer-Verlag.

Kowalski, R. A. and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

Luccarini, L., G. L. Bragadin, G. Colombini, M. Mancini, P. Mello, M. Montali, and D. Sottara. Formal verification of wastewater treatment processes using events detected from continuous signals by means of artificial neural networks. Case study: SBR plant. *Environmental Modelling & Software*, 25(5):648–660, 2010.

Marsili-Libelli, S. Control of SBR switching by fuzzy pattern recognition. *Water Research*, 40(5):1095–1107, March 2006. PMID: 16494923.

McCarthy, J. and P. J. Hayes. Some Philosophical Problems From the StandPoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.

Sottara, D., A. Manservisi, P. Mello, G. Colombini, and L. Luccarini. A cep-based soa for the management of wastewater treatment plants. In *Environmental, Energy, and Structural Monitoring Systems, 2009. EESMS 2009. IEEE Workshop on*, pages 58–65. IEEE, 2009.

Traore, a., S. Grieu, S. Puig, L. Corominas, F. Thiery, M. Polit, and J. Colprim. Fuzzy control of dissolved oxygen in a sequencing batch reactor pilot plant. *Chemical Engineering Journal*, 111(1):13–19, 2005.