Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 104, Issue 18, March/April 2019, Pages 23–46
https://doi.org/10.7250/csimq.2019-18.02

# Extending Business Process Models with Appreciation

Irina Rychkova[1*], Gil Regev[2], and Alain Wegmann[2]

[1]University Paris 1 Pantheon-Sorbonne, 12 Place du Panthéon, Paris 75005, France
[2]Ecole Polytechnique Fédérale de Lausanne (EPFL), School of Computer and Communication Sciences, CH-1015 Lausanne, Switzerland

irina.rychkova@gmail.com, gil.regev@epfl.ch, alain.wegmann@epfl.ch

**Abstract.** We use homeostasis, the maintenance of steady states in an organism, to explain some of the decisions made by participants in a business process. We use Vickers' Appreciative System to model the homeostatic states with Harel's statecharts. We take the example of a doctoral student recruitment process formally defined between a faculty member, a graduate student candidate and a doctoral school. We analyze some gaps in the process caused by a misfit between norms of the process participants. We present a rationale for the anticipation and resolution of these misfits. We extend the traditional operational model with an appreciative model. This model represents the appreciative systems of process participants. Understanding these appreciative systems is necessary to make explicit the misfit between the model and the observed reality. The operational model represents the "technical" perspective on the business process, the one that can be automated. The appreciative model represents the "social" perspective, the one that explains the participants' behavior as a result of their individual and collective norms. By combining these two perspectives, we can appreciate the richness of the development of socio-technical systems.
**Keywords:** Appreciative System, Statechart, Business Process, Social Perspective, Decision Making.

## 1 Introduction

Business process models treat all participating actors alike, as deterministic machines. Human actors are seen as no different than automats producing results based on predefined inputs. The psychological and social dimensions that characterize human behavior are not represented in these models. In this article, we explore how these dimensions may be added to a traditional business process model. We use Vickers' appreciative system [1], [2], [3] as a model of human behavior. Vickers created this model to show how people made sense of themselves and their environment. Vickers was feeling that models based on goal achievement did not give a good description of the richness of human interpretation of complicated situations.

[*] Corresponding author

Reference: I. Rychkova, G. Regev, and A. Wegmann, "Extending Business Process Models with Appreciation," Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 18, pp. 23–46, 2019. Available: https://doi.org/10.7250/csimq.2019-18.02

Additional information. Author's ORCID iD: I. Rychkova – https://orcid.org/0000-0002-1100-0116. PII S225599221900104X. Received: 1 January 2018. Accepted: 8 February 2019. Available online: 30 April 2019.

We explain homeostasis [4] and Vickers' appreciative system, and show how these concepts can be applied to business process modeling. We use the example of a recruitment process in which graduate student candidates are evaluated to be hired as doctoral students. We use the term PhD as an abbreviation of "doctoral student." We use statecharts to model the process as defined by the university. This process includes the Doctoral School, a Faculty Member and a Graduate Student (a candidate). We first create a model that we call *operational*. It shows the standard process model that is usually drawn to automate behavior. We then discuss that the operational model captures the "technical" perspective of the process keeping the "social" perspective of the process participants implicit. We make this social perspective explicit with Vickers' appreciative system in what we call the *appreciative model*. This model exposes the social norms that govern the behavior of the process actors.

This is a preliminary work that extends our previous publication at STPIS'18 workshop [5]. With this work we would like to show that it is feasible to model an appreciative system with statecharts, to link it to a business process model, and therefore to enlarge the technical scope of business process modeling with a social perspective.

In Section 2 we introduce the example of the doctoral student recruitment process and provide the basic concepts of statecharts. In Section 3 we present the operational model of the doctoral student recruitment process as defined in the university. In Section 4 we present Cannon's framework of homeostasis and Vickers' appreciative system and propose the formalization of an appreciative systems with Harel's statecharts. We extend the operational model of the doctoral student recruitment process with an appreciative model in Section 5. In Section 6, we discuss the interest and some challenges of formalizing the appreciative systems. In Section 7 we address the future work and conclusions.

## 2 The Running Example and Introduction to State Charts

This section describes the principle of state chart models that we will use to describe the operational model as well as the appreciative model. It also introduces the example we will use to illustrate the concepts we propose to include in process modeling.

### 2.1 Example: The PhD Recruitment Process

In this work we use an example of a PhD recruitment process of a University. The process involves three actors: a faculty member (FM) – a professor leading a research group and recruiting a doctoral student for the team; a graduate student (GS) searching for a PhD position; and a doctoral school (DS) that preselects the PhD candidates and manages them until their graduation. This process is a simplified view of the PhD recruitment processes experienced by the authors of this article. It omits other actors (e.g. the HR department or the dean of the university who might be also involved).

In order to be recruited as a doctoral student by the university, a GS

- has to be accepted by the DS and
- has to be approved by a prospective thesis supervisor (a FM).

The GS can start the process either by sending an application to the DS or by directly contacting the FM whose team she/he aims to join. In the first case, once accepted, GS needs to apply for the positions with different FMs. In the second case, once approved, the GS needs to apply to the DS for evaluation.

The DS receives and evaluates the GS' applications by examining their academic background and track of records. If the DS accepts the GS candidate, the candidate needs to search for a thesis supervisor by sending applications to various FMs. If no supervisor is found within some predefined time frame, the GS cannot be recruited and the process stops.

The FM evaluates GS applications mainly focusing on the candidate's motivation and skills related to the specific research topic and can approve or reject the GS. If the FM approves the graduate student candidate, the latter must also be accepted by the DS. If DS rejects the graduate student candidate, the GS cannot be recruited and the process stops.

This double stage process is designed in order to guarantee the excellence of graduate student candidates, evaluate their capacity to integrate in a university doctoral school program and in a specific research team and, also to successfully finish their training by obtaining the PhD degree.

## 2.2 Statecharts and YAKINDU SCT

**Statecharts.** We model the PhD recruitment process using statecharts defined by Harel [6]. Compared to a workflow paradigm widely used for process modeling, we specify the process with a set of states and transitions between states. This allows us to omit specification of concrete activities associated with recruitment but to focus on the process goals and milestones.

Thus, statecharts provide a uniform modeling notation to reason about the operational (business) process as specified between the process participants and about the cognitive process associated with decision-making of each participant.

**Statecharts formalism**. The statecharts formalism specifies a hierarchical state machine (HSM) that extends classical finite state machine (FSM) [7] by providing:

- Hierarchy (or depth) – the possibility to model states at multiple hierarchical levels, with the notion of abstraction/refinement between levels;
- Orthogonality – the possibility to model concurrent or independent submachines within one state machine;
- Broadcast communication – the possibility to synchronize multiple concurrent submachines via events.

As for FSM, states and transitions are central concepts in statecharts. State specifies a state of the modeled system. A *state* can have a behavior that describes which actions are taken under which conditions. *Transitions* between states are triggered by events. If a transition t takes place only if an event e occurs – e is called a *triggering event* of the transition t. Triggering events can be complemented with a guard conditions: *e[c]*. In this case, we say that a transition t takes place when e occurs and c holds.

During an execution of a state machine, a state can be active or passive. Due to a hierarchical structure, where several (sub)states can be embedded in a (parent) state, a multiple states of a statecharts can be activated at a given moment of execution. These states are called an *active configuration* of the state machine. Compared to a FSM transition, a statechart transition can be seen as a change from one active configuration to another. We illustrate this formalism and the notion of model execution on the examples in the following sections.

**Yakindu.** We use YAKINDU Statechart Tools (YAKINDU SCT) [8] for modeling, simulation and analysis of the process. YAKINDU is a modular toolkit for developing, simulating, and generating executable finite-state machines (FSM). YAKINDU statecharts are organized in regions. Hence it is possible to organize multiple state machines in different regions and to run them concurrently. Regions contain states and transitions. A state, in turn, can contain one or more regions, turning it into a composite state (contains one region) or an orthogonal state (contains two or more regions).

A state can define behavior in the form of one or several trigger [guard] / effect statements. These statements provide a declarative (non-prescriptive) specification of behavior as no execution order is applied.

*Triggers* are events that can be complemented with a guard condition (*guard*). The effect will only be executed if the trigger occurs and if the guard condition holds. The *effect* can include one or several actions such as assigning a value to a variable, raising an event, calling a function.

We simulate the statechart models with Simulation view in YAKINDU STC. The framework allows us to manually raise events, to inspect and modify variables of a running simulation and to observe the model's behavior as a sequence of active configurations triggered. We use this simulation tool to play and discover different process scenarios. More advanced features for model simulation are available in YAKINDU. They are out of the scope for this article.

## 3 The Operational Model of the PhD Recruitment Process

The PhD recruitment process, as defined by the University, provides the details on the steps that must be taken starting from the point when a graduate student (GS) applies for a PhD. This application can be done either by sending an application to the doctoral school of the university (DS) or by contacting a professor (FM). The PhD recruitment process terminates with a recruitment or rejection of the GS. In this section we model the process of the PhD recruitment described above with statecharts using YAKINDU STC. We explain the model and how it can be simulated with YAKINDU. We call this process model "operational" as opposed to the "social" process model discussed in the next sections.

### 3.1 Using statecharts for process modeling

We model the process as collaboration between its three participants. We use statecharts to describe the role of each participant in the process. Each statechart contains states and transitions describing the behavior of the corresponding participant and is illustrated in: Figure 1 (Graduate Student), Figure 2 (Doctoral School), and Figure 3 (Faculty Member). The process model (Figure 5) shows collaboration between participants and is organized in three independent regions.

Process participants exhibit concurrent behavior and communicate via messages.

The statecharts diagram in Figure 5 is also a statechart and can be used as a part of a more complex diagram. This creates a hierarchy of statecharts. Hierarchical statecharts are successfully applied for modeling complex real-time systems [6] and thus we consider that they can also be applicable for real business processes.

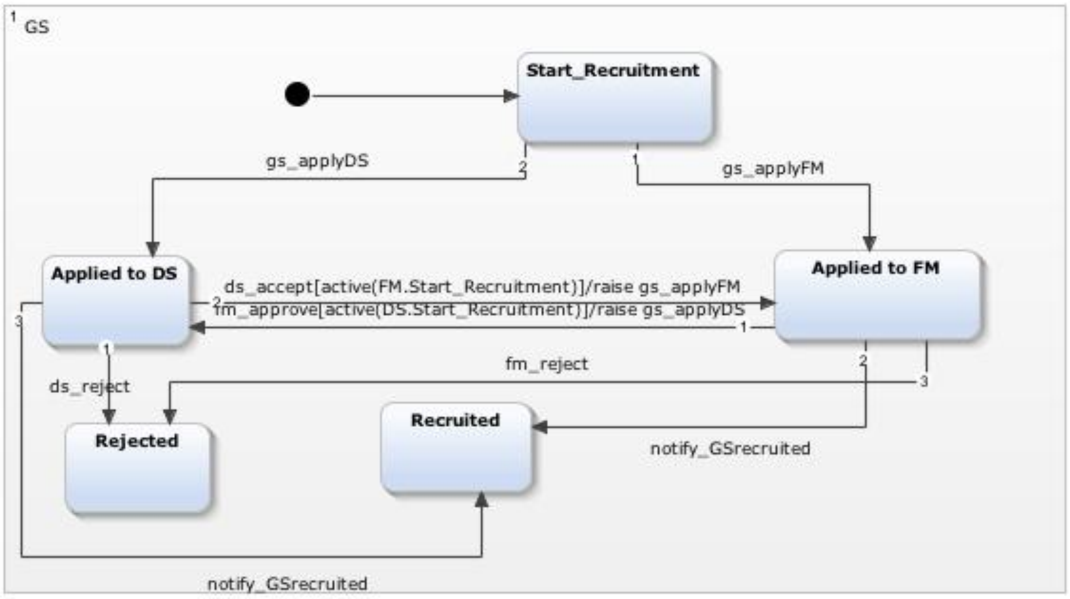


**Figure 1.** Statecharts model of behavior of the Graduate Student (GS) within the formal PhD recruitment process defined by the University.

The Graduate Student “GS” statechart (Figure 1) describes a behavior of a graduate student who applies for a PhD position. The statechart is specified with five states: Start_Recruitment (the default starting state), Applied to DS, Applied to FM, Rejected and Recruited states.

State transitions (depicted by the arrows between states) are triggered when their corresponding triggering conditions are satisfied. The triggering conditions are modeled with “**condition[guard]/effect”** statements. The condition is specified with a boolean expression or an event raised by some statechart (another process participant in our case). It can be guarded by an invariant. The effect describes what will happen along with the transition (e.g. an event can be raised or an operation can be executed) and is optional. The triggering conditions are shown next to their corresponding transitions (the arrows) in the model. For instance, the transition “**Applied_to_DS → Rejected”** in GS (Figure 1) is triggered when the **ds_reject** event is raised (no guard is specified). We model the events with a prefix indicating the statechart raising this event: **ds_** corresponds to the DS statechart (represented in Figure 2).

The transition “**Applied_to_DS → Applied_to_FM**” in GS (Figure 1) refers to the case when the graduate student, once accepted by the DS, has to apply to a faculty member (or FM). The statement on the transition *ds_accept[**active**(FM.Start_Recruitment)]/**raise** gs_applyFM* reads as follows: when the DS accepts the graduate student candidate (ds_accept event is received) and the graduate student candidate is not yet reviewed by a FM (the FM statechart is in its starting state: active(FM.Start_Recruitment)), the graduate student applies to a FM (the event gs_applyFM is raised). This event triggers a transition in the FM state chart (Figure 3).

The transition “**Applied_to_FM → Applied_to_DS**” in GS (Figure 1) refers to the opposite case – when the graduate student gets approved by a FM (i.e. fm_approved is received) while not yet applied to the DS.

The Doctoral School “DS” statechart (Figure 2) describes a behavior of the doctoral school. The statechart is specified with five states: Start_Recruitment (the default starting state), Candidate_Review, Accepted, Rejected and Recruitment.

The transition “**Candidate_Review → Accepted**” in Figure 2 refers to the case when the DS accepts a graduate student while the candidate is not yet approved by a faculty member (thus the DS waits for the decision to terminate the process). The statement on the transition ds_accept[!**active**(FM.Approved)] expresses this behavior.

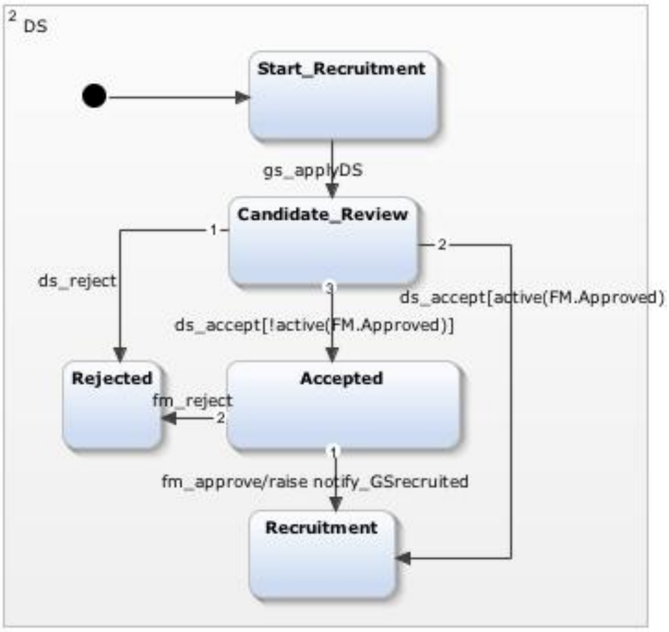


**Figure 2**. Statechart model of behavior of the Doctoral School (DS) within the formal PhD recruitment process defined by the University

The transition "**Accepted → Recruitment**" in Figure 2 refers to the case when for a graduate student accepted by the DS (being in the state Accepted), the DS eventually receives an approval from a FM. From the point of view of the doctoral school, the recruitment conditions are fulfilled, the procedures can be now finalized and the GS can be notified on recruitment. The statement on the transition *fm_approve/**raise** notify_GSrecruited* expresses this behavior.

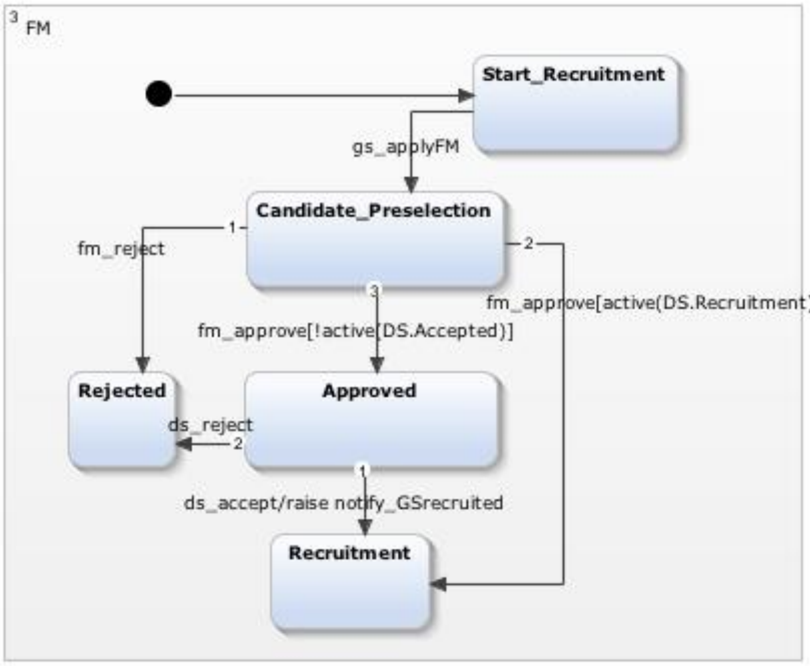


**Figure 3.** Statechart model of behavior of the Faculty Member (FM) within the operational PhD recruitment process defined by the University

The Faculty Member "FM" statechart (Figure 3) describes a behavior of the faculty member. The statechart is specified with five states: Start_Recruitment (the default starting state), Candidate_Preselection, Approved, Rejected, and Recruitment. The transition logic is similar to one described above for the DS.

## 3.2 Interaction Between Process Participants

The DS, FM, GS statecharts are running concurrently: each statechart represents an actor who reacts on the events raised by the other actors (statecharts) in the model. In Figure 4 we present a list of events defined for the operational PhD recruitment process.

```
internal:
// operational process events
// user events (manuall):
            event gs_applyDS
            event gs_applyFM
            event fm_approve
            event fm_reject
            event ds_accept
            event ds_reject
// simulated events:
            event notify_GSrecruited
```

**Figure 4.** Events defined for the Operational PhD recruitment process

All the events except of the last one are specified with the prefix (ds_, fm_ or gs_) that identifies the "sender" of the event. The last event "notify_GSrecruited" can be raised by either the DS or FM, when the conditions for recruitment are fulfilled.

Events represent the observed results of some activities during the process. During the simulation, all the events specified in the model are broadcasted and every statechart in the model "receives" them or "sees" them occur. However, it is a modeler's choice to specify which events will be ignored and on which events the statechart will react and how. The modeler can specify a behavior that can be triggered (a) upon receiving some event when entering, exiting or resting at some state or (b) when a transition between states is triggered.

In the model, the corresponding behavior statements and their triggering conditions can be shown **on state transitions** (see the examples above) or **inside the states** (this will be explained in the next section).

## 3.3 Operational Model Simulation

Model simulation in YAKINDU allows for visualizing different process scenarios. During the simulation, the simulation window shows the current active state of each statechart and the last triggered transitions (they are highlighted in color in Figure 5 (a)). This is called "current active configuration" of the statechart. The simulation window also provides the list of events (Figure 5 (b)) that can be raised during the simulation. In the simple simulation mode, the user clicks on these events, one at a time, simulating their occurrence and the statechart reacts by changing the active configuration. As discussed in Section 3.2, some events can be raised by statecharts as a part of their behavior expression (specified by the "**raise**" keyword). The simulation terminates when some final state or final configuration is reached. More complex simulation can be done with YAKINDU, when different scenarios can be preconfigured and played automatically or driven by some program at the backend.

In our example, it is the user who simulates the event occurrence and can decide upon their order. Note that events can have an effect in current active configuration only if at least one behavior statement describing the reaction on the event is specified in this configuration. Conversely, if no behavior that describes a reaction on an event is specified – the event will be ignored by the statechart. For instance, consider the current active configuration as shown in Figure 5. If the user clicks "notify_GSrecruited" event from the list – its occurrence will have no effect and the active configuration will not change. Clicking "ds_reject," however, will immediately trigger transitions in both DS and GS statecharts and the active configuration will change.

In our example, the user plays the roles of different process participants and thus has to respect some logic in order of events that can be raised by these participants. Thus, when the model simulation starts, the user plays the role of the graduate student (GS) by applying either to the DS or to the FM; next it has to simulate the corresponding decision of the DS or FM and so on.

Figure 5 (a) illustrates the configuration of the statechart after the user raises gs_applyDS event from the simulation window (by clicking gs_applyDS): here the GS statechart passes from Start_Recruitment to Applied to DS state, DS passes from Start_Recruitment to Candidate_Review and FM remains in the Start_Recruitment state. From this configuration, the next step will be to "simulate" the decision made by DS on the application by raising ds_accept or ds_reject events (i.e. the user needs to choose and click on the corresponding event in the list illustrated in Figure 5 (b)). Here the user plays the role of the doctoral school (DS) in the process.

While occurrence of major events in our model is simulated by the user, some events are automatically raised: the transition "Accepted → Recruitment" of the DS statechart in Figure 5 (a) is specified with the expression *fm_approve/**raise** notify_GSrecruited*. Here the fm_approve event need to be raised by the user (clicking in the simulation window), but the notify_GSrecruited event will be raised automatically as a part of the behavior expression specified for the transition.

Example of scenarios to play with this model:
**gs_applyDS – ds_accept – fm_approve – notify_GSrecruited** – here the graduate student candidate applies to the doctoral school, gets accepted, applies to a faculty member, gets approved and eventually recruited.
**gs_applyFM – fm_approve – ds_reject** – here the graduate student candidate applies to a faculty member, gets approved, applies to the doctoral school, but gets rejected.
**gs_applyDS – gs_applyFM – fm_approve – ds_approve – notify_GSrecruited** – here the graduate student candidate applies to the doctoral school and to a faculty member (not waiting for being accepted), gets accepted, gets approved and eventually recruited.
Other variants of scenarios are possible.

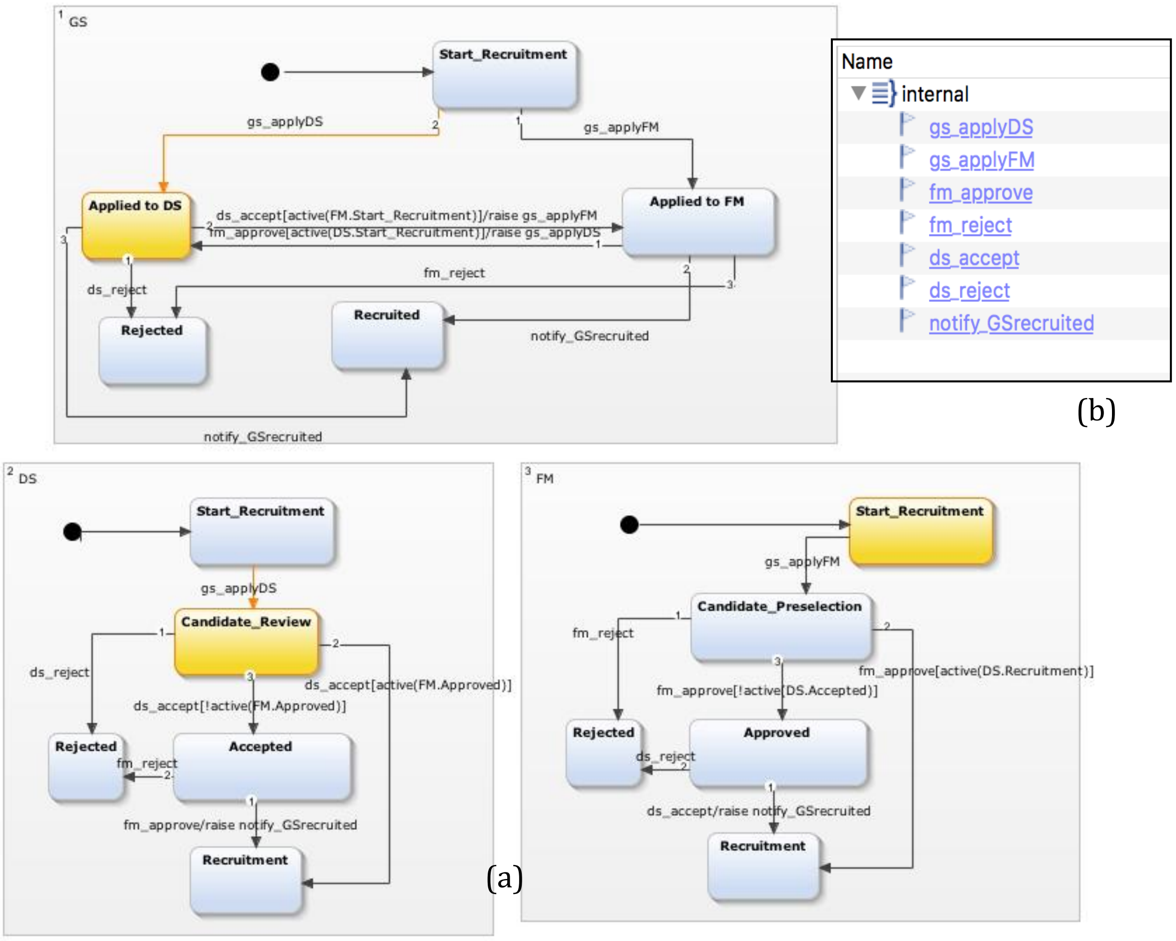


**Figure 5.** Simulation of the operational model with YAKINDU: (a) represents the three concurrent state machines and (b) shows the events sent by the person running the simulation (i.e. the actual decisions taken by the three actors).

## 3.4 Discussion on the Process Efficiency – Implicit Gap

The graduate student candidate is recruited when two conditions are fulfilled: FM approves the candidate and DS accepts the candidate. It is known from practice that the DS evaluates the academic background of the graduate student candidate. FM evaluates GS' applications mostly focusing on candidate's motivation and skills related to the specific research topic. This double stages process is designed to ensure the graduate student candidate excellence both on the university level and on the level of the research team.

An assumption is made that an academic excellence of a GS as defined by the doctoral school is a necessary condition for successful integration of the GS in the research team and completion of the PhD training. Nevertheless, some faculty members do struggle to hire a PhD student within this process. This creates a gap (a conflict) in the organization. The gap is implicit in the model.

The concrete evaluation (or ***how*** the DS and FM are making their respective decisions about graduate student candidate excellence) is implicit in the model. In order to understand the gap and to eventually improve the process, we propose to examine the appreciative systems (as proposed by Vickers) of the FM and DS. These appreciative systems will be formally represented in one appreciative model per actor. In the next section, we discuss the theories of homeostasis [4] and appreciative system [1], [2], [3]. We then model the appreciative systems of the DS and FM with statecharts, creating a **social** model of the PhD recruitment process. Compared to the **operational** model discussed above, the social model is focused on the cognitive aspect of the recruitment: it shows how this process is perceived by the FM and DS and how the decisions regarding the graduate student candidate excellence are made by these participants.

## 4 From the Technical Perspective to the Social Perspective of the Business Process: Appreciative Systems

### 4.1 Homeostasis and Appreciative Systems

A business process is often associated with achievement of some or several well-defined goals. This can be seen as the direct implementation of Cybernetics as defined by Rosenblueth, Wiener and Bigelow [9]. Rosenblueth was a collaborator of Cannon [10] who, in the 1920s, coined the term Homeostasis in order to explain how an animal body maintains the steady states that are the basis of its survival [4], [11]. Cannon explained that living organisms somehow found a way to maintain steady states even though they are made of unstable internal elements and live in an unstable external environment. It is the maintenance of these more or less stable internal states that maintain a living being's identity and therefore its survival.

In Cannon's work, as explained by [12], there is no goal to be achieved, just the maintenance of steady states. Rosenblueth, Wiener and Bigelow simplified Cannon's work and defined teleological, purposeful, behavior as achieving a well-defined final state through the use of a negative feedback mechanism [10]. Whereas the early work in Cybernetics involved the study of man-made systems, it was very quickly applied to socio-technical systems. Business process management is one such example. The goals that are to be achieved by business processes are the modern-day descendants of this early teleological work.

Writing from a social perspective, Vickers [1], [2], [3] took the work created in Cybernetics [9] and re-expanded it with maintenance in mind, but this time writing about the maintenance of relationships instead of states. The maintenance of a relationship is, in fact, the maintenance of a relationship in a given state. Vickers wrote about attaining, maintaining and eluding relationships [11]. Maintaining and eluding a relationship can both be seen as keeping it in a specific state, either close or distant.

In Vickers' work these relationships are maintained with respect to norms, states that remain more or less the same over time, just like Homeostasis is the maintenance of (quasi) stable states. For Vickers, norm-holding was very different from what he termed goal-seeking (goal-seeking is known today as goal achievement in Requirements Engineering [13], [14]). Achievement goals are met once and for all, and determine a well-defined end point. For example, to accept or reject a PhD candidate are two end states of the PhD recruitment process. Norm holding defines an on-going activity of matching the current state of affairs with the relevant desired state [11]. Accepting or rejecting a PhD candidates are only steps in the overall on-going activities of a DS and FM in which these structures must be kept alive by regulating mutual relationships. In Requirements Engineering, the concept of maintenance goal was suggested to represent this on-going activity [13], [14]. The concept of maintenance goals was defined earlier on in [15] and [16].

The desired state can be described with a set of steady states called norms. Each norm is associated with a threshold, (e.g. a human body temperature is maintained at approximately 37°C, with an associated threshold of a fraction of degree). An action is taken whenever the actual state is sensed to be outside of the threshold. This well-known process of regulation has no beginning or end. It operates as long as the subject survives and is responsible for its survival. This results in a homeostatic state [4].

Unlike the animal or machine, in which the norms are often pre-defined by either evolution or design, the norms that are held by an organization are the result of the on-going process of appreciation. They are therefore generated and maintained by the same process they govern and should not be considered as immutable or given by a designer [3].

Vickers distinguished two "segments" in the regulation process. In the first segment, the current state is compared with the norm. The resulting information is fed to the second segment in which an action is taken to correct the state if it is outside of the threshold associated with the norm [3]. Vickers notes that the second segment has received much more attention in system engineering and problem solving than the first segment. In policy-making, he observes, the major difficulty is in the first segment. In this segment the challenge is to define the criteria for capturing the current state and evaluating it against multiple potentially inconsistent norms [3]. It is this attention to the first segment that most distinguishes norm-holding from goal-seeking. Vickers conceived the appreciative system, a model of policy-making as norm-holding, to give more attention to the first segment.

Vickers' appreciative system is composed of 3 distinct, but interrelated elements [2], [3], [17]: Reality Judgments (RJ), Value Judgments (VJ) and Action Judgments (AJ). Reality judgements and value judgments can be seen as comprising the first segment. Action judgments can be seen as belonging to the second segment.

Reality judgments correspond to what people perceive of their situation. Value judgments correspond to the selection of the norms relevant to these reality judgments and how the actor compares the reality judgments with these norms, resulting in a realization that the situation is acceptable or not. Vickers further detailed the stages of the Value Judgment component as [17]:

- Attaching a Reality Judgment to an existing category and thereby defining the relevant norm (Matching).
- Evaluating the Reality Judgment (the state of affairs) on present and future relations with the help of the norm (Weighing).
- Creating a new category for future exercises of the appreciative system (Innovating).

Action judgment is prompted if the situation is unacceptable in order to bring it to a more acceptable form. The action can be directed inwards (in order to change the "inside" of the system: norms, value judgments, reality judgments), outwards (in order to change the "outside" of the system, it's environment) and can be "not to change anything." Thus, action judgments correspond to the choice of relevant behavior (activities) within the selected response strategy.

Vickers describes the overall functioning of the appreciative system as a process of optimizing and balancing [3], [17]. Optimizing is geared toward what Vickers calls functional (or service) relationships. It attempts to keep the relationships with the actor's external entities in a relatively good state (akin to Simon's satisficing concept) [1]. Balancing tries to prevent the depletion of its resources; which Vickers calls metabolic relationships. The maintenance of the requisite functional relationships is dependent on the metabolic relationships and vice versa. Thus, the appreciative system continually tries to provide as good a service as possible to its external stakeholders (optimizing), with its available or obtainable resources (balancing the inputs and outputs) [3].

The repeated exercise of the appreciative system leads to what Vickers calls readiness in each of its components: Readiness to see (related to RJ), Readiness to value (related to VJ) and

Readiness to act (related to AJ). At any given moment this readiness defines a specific state of the appreciative system, which Vickers calls the appreciative setting [1].

## 4.2 Modeling Appreciative Systems

In this section we describe a generic appreciative model that shows how the concepts defined by Vickers (i.e. norm, threshold, reality judgment, value judgment, and action judgment) can be modeled with statecharts for a generic actor. In the next section, we use this generic model to define appreciative systems for the faculty members and the doctoral school actors in our PhD recruitment process.

The Appreciation model, is illustrated in Figure 6, consists of one composed state called A_Setting. A_Setting represents the appreciative setting of the actor and can be associated with one norm or the set of norms N the actor seeks to *maintain (i.e. keep unchanged)*. Each norm N is associated with a tolerance threshold. When this threshold is crossed, the system is not in its homeostatic state and has to respond to regain it by acting on the environment or on its appreciative setting.

We represent Reality Judgments (RJ), Value Judgments (VJ) and Action Judgments (AJ) as sub-states of the A-Setting state. The regulation process is captured with states transitions. For each of the states, we define a behavior that will be triggered upon entering, exiting or while remaining in this state with a list of *trigger[guard]/effect* statements. Figure 7 depicts the list of variables and events that can be captured or raised by the appreciative model. These events and variables are used in the *trigger*, *guard* or *effect* expressions of the behavior specification.

### Step 1: Actor's Reality Judgment

RJ is the default starting state for the regulative process. In the RJ state, the actor captures the events e (Figure 6) raised by the environment (i.e. another actor) and transforms them into reality judgments. We do not show the details of this transformation. These reality judgments are further used in the VJ state for the situation assessment, a transition toVj (to the VJ state) is triggered.

### Step 2: Actor's Value Judgment

In the VJ state, the actor evaluates the situation by selecting relevant norms to compare with the reality judgments r1, r2, and concluding if their corresponding thresholds are crossed or not. Thus, each ri can take a value of "OK" or "NOK."

Still in Figure 6, we model value judgments with the events: r1isOK, r1isNOK, r2isOK, r2isNOK. Simulating the value judgment in YAKINDU, r1isOK will be raised when r1 is considered within the threshold and r1isNOK will be raised otherwise. In the current model, value judgments must be raised by the user from the simulator view. Potentially this can be automated with YAKINDU.

Each value judgment automatically triggers a transition to the AJ state (toAj event is raised automatically, see Figure 7). In order to make more than one value judgment, the user will need to raise the addVj event in the AJ state.

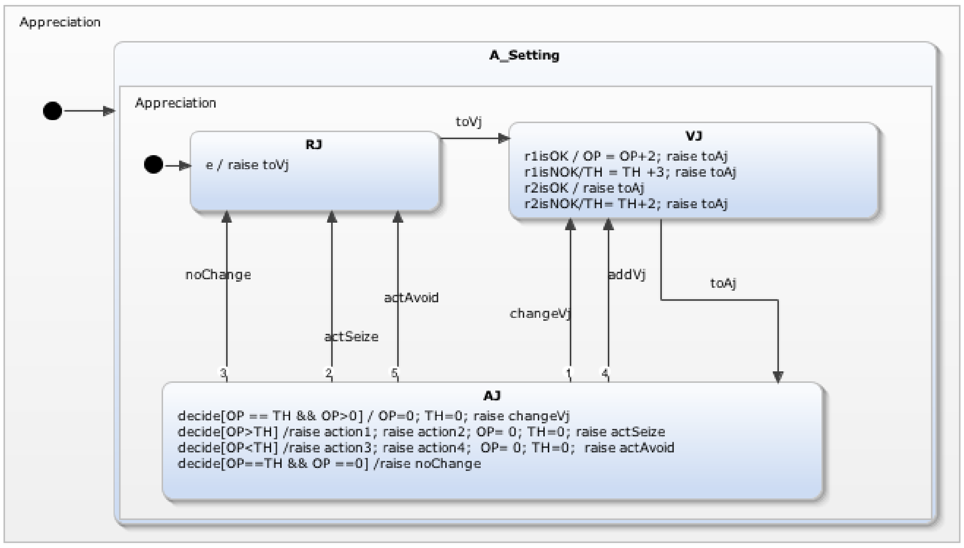

Appreciation
A_Setting
Appreciation
toVj
RJ
e / raise toVj
VJ
r1isOK / OP = OP+2; raise toAj
r1isNOK/TH = TH +3; raise toAj
r2isOK / raise toAj
r2isNOK/TH= TH+2; raise toAj
noChange
actAvoid
actSeize
changeVj
addVj
toAj
3
2
5
1
4
AJ
decide[OP == TH && OP>0] / OP=0; TH=0; raise changeVj
decide[OP>TH] /raise action1; raise action2; OP= 0; TH=0; raise actSeize
decide[OP<TH] /raise action3; raise action4; OP= 0; TH=0; raise actAvoid
decide[OP==TH && OP ==0] /raise noChange

**Figure 6**. A generic appreciative model

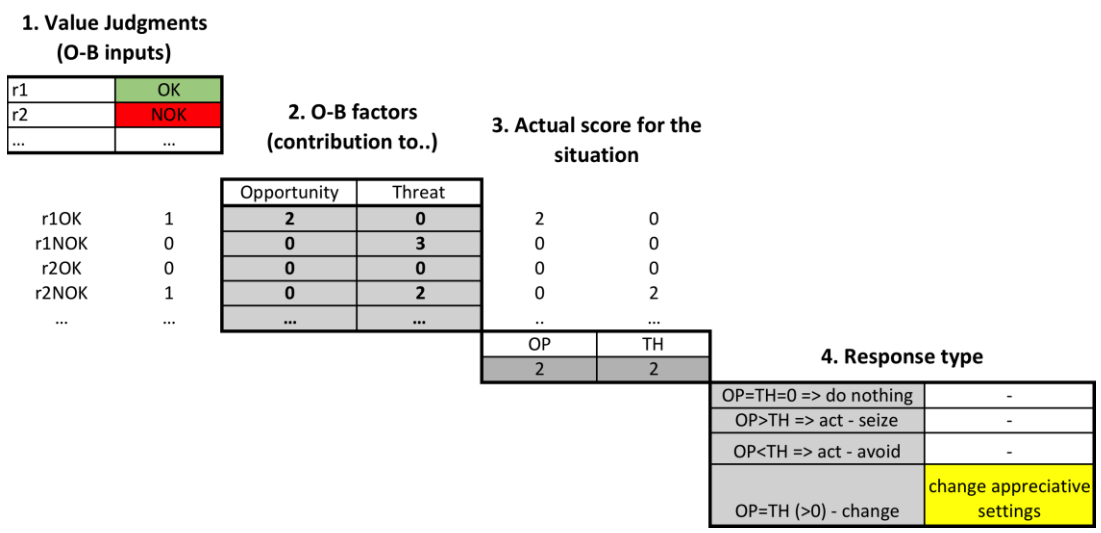
## Optimizing and balancing

```
internal:
            event toVj                    //-automatic, raised when RJ is created
            event toAj                    //-automatic, raised when VJ is made
            event addVj                   //-raised by the used to make a new VJ
            event decision                //-raised by the user when decision making is
finished
// response types  - automatic, based on optimizing-balancing
            event noChange
            event actSeize
            event actAvoid
            event changeVj
//value judgments             - manual, raised by the user
            event r1isOK
            event r1isNOK
//Optimizing-balancing
            var OP: integer
            var TH: integer
//observations (inputs) –user events (manual)
            event e
//actions affecting the reality (outputs) – simulated events
            event action1
            event action2
            event action3
            event action4
```

**Figure 7.** Events and variables used in the Appreciation statechart

We define optimizing-balancing as a function F(r1,...) that transforms any combination of value judgments {.. riOK, rjNOK, ..} into a pair {OP, TH}, where OP indicates how much the actor perceives the overall situation as an **opportunity** (something we wish to seize or take) and TH indicates how much the actor perceives this situation as a **threat** (something that we wish to avoid). This perception of the situation as a whole defines the type of response the actor eventually produces.

In Figure 8, we illustrate optimizing-balancing. Once a value judgment is made, we use the table of optimizing-balancing that indicates "how much" this value judgment is an opportunity and how much it is a threat. Summing up the obtained scores for all value judgments made about the situation we obtain OP and TH respectively. We call these the "optimizing-balancing factors."


**1. Value Judgments (O-B inputs)**

| | |
|---|---|
| r1 | OK |
| r2 | NOK |
| … | … |

**2. O-B factors (contribution to..)** / **3. Actual score for the situation**

| | | Opportunity | Threat | | |
|---|---|---|---|---|---|
| r1OK | 1 | **2** | **0** | 2 | 0 |
| r1NOK | 0 | **0** | **3** | 0 | 0 |
| r2OK | 0 | **0** | **0** | 0 | 0 |
| r2NOK | 1 | **0** | **2** | 0 | 2 |
| … | … | **…** | **…** | .. | … |

| OP | TH |
|---|---|
| 2 | 2 |

**4. Response type**

| | |
|---|---|
| OP=TH=0 => do nothing | - |
| OP>TH => act - seize | - |
| OP<TH => act - avoid | - |
| OP=TH (>0) - change | change appreciative settings |

**Figure 8.** Optimizing-balancing

We distinguish several response types based on the total OP-TH scores.

When OP ≠ TH – the response will be directed *outwards* (in order to change the system's environment, producing some impact that can be captured by external actors). In particular, when OP > TH – the actor will act to seize what it sees as an opportunity; when OP < TH – the actor will act to avoid what it sees as a threat.

When OP == TH == 0 – the observed situation does represent neither a threat nor an opportunity and the response will be *neutral* ("not to change anything").

When OP == TH ≠ 0 – the situation is unclear for the actor, and the response will be directed *inwards* (in order to change the appreciative setting, i.e. norms, value judgments, reality judgments, optimizing and balancing factors).

In the example above, if we change the optimizing-balancing factors for r1OK from [2,0] to [3,0] , we will obtain OP = 3 and TH = 2 for the situation. This will result in OP > TH– the *act-seize* response type.

In the appreciative model in Figure 6, the optimizing-balancing is shown in the VJ state:

```
r1isOK / OP = OP+2; raise toAj
r1isNOK/TH = TH +3; raise toAj
r2isOK / raise toAj
r2isNOK/TH= TH+2; raise toAj
```

Raise toAj stands for a transition to the AJ state.

To reflect the real cognitive process of evaluation, we assume that any number of value judgments can be considered. During simulation, addVj events can be raised many times in order to mimic a human process of appreciation: one individual may undertake a thorough reflection cycle VJ→AJ→VJ→..., with various assessments concurrently increasing and decreasing OP and TH, until the moment she is satisfied with her assessment. And conversely, an individual can make a hasty conclusion, making a judgment based on one parameter only and passing to action: VJ→AJ

## Step 3: Actor's Action Judgment

Action judgments correspond to the choice of relevant behavior (concrete actions) corresponding to the selected response type. In the statechart in Figure 6 we model it with the following statements:

```
decide[OP == TH && OP>0] / OP=0; TH=0; raise changeVj
decide[OP>TH] /raise action1; raise action2; OP= 0; TH=0; raise actSeize
decide[OP<TH] /raise action3; raise action4;  OP= 0; TH=0;  raise actAvoid
decide[OP==TH && OP ==0] /raise noChange
```

During the simulation, when in the AJ state, the user raises the event *decide* that stands for executing a response. We use OP and TH variables in a guard expression to select the right response that matches the guard.

**changeVj** response brings the statechart back to the VJ state and resets OP and TH values. – here we simulate the change of value judgments. In the current model we can illustrate only this type of “inward” changes or changes in appreciative settings during the simulation since the other settings cannot be changed “on the fly” and require modification of the statechart.

**actSeize** and **actAvoid** responses correspond to the “outward” response, where some actions (e.g. action1, action2, ...) will be executed and TH and OP will be reset. These actions define how exactly the actor will interact with the environment (i.e. other actors) in order to seize the opportunity or to avoid the threat respectively. This simulates a change in the environment that results in a change in RJ. The limitations of our model make it a closed-loop but in reality the action is made on the environment and then sampled again in the RJ. The Appreciation statechart will be set to its initial RJ state for the next cycle of the regulative process.

**noChange** response resets the Appreciation statechart bringing it to its initial RJ state without any change (to the environment or to the appreciative settings).

## 5 Extending the Operational Model with the Model of Appreciative Systems for the PhD Recruitment Process

We extend the model of the operational PhD recruitment process presented in Section 3 with the appreciative models of the Faculty Member (FM) and Doctoral School (DS). We examine the reasons for the issue observed in the operational process – the fact that some faculty members struggle in order to recruit a PhD student within this process.

To formalize the regulative processes that support decision making for PhD recruitment, we define two statecharts: DS_ (Figure 9) and FM_ (Figure 11). These statecharts represent the appreciative systems of the Doctoral School (DS) and the Faculty Member (FM) respectively. They instantiate the model explained in the previous section. We add these statecharts to the initial model available (in Figure 1) as two separate regions. This forms the extended model that is illustrated in Figure 13.

We present the list of events defined for DS and FM appreciative system models in Figure 14. The events defined for the operational model (Figure 4) remain unchanged. However, now we can distinguish between the events raised in the operational model and used as *observations* in the appreciative systems (gs_applyDS, gs_applyFM). The same applies to the *responses* raised in the appreciative models (Figure 3) and used as *decisions* in the operational model (ds_accept, ds_reject, fm_approve, fm_reject).

The appreciative models explain how these observations are transformed into responses.

### 5.1 Norms and Thresholds of the DS and FM

We begin by identifying the norms that affect the decision to accept a graduate student candidate by the DS and the decision to approve the graduate student candidate by the FM.

For the doctoral school (DS) the norm can be formulated as follows: DS_Norm = “*To ensure strong background of a* graduate *student candidate in the research areas defined by the school* “

We define one aggregate variable v that corresponds to this norm and allows for assessing the graduate student candidate excellence. As an example, this variable can be the graduate student candidate’s GPA (Grade Point Average), or a score for some specific modules (Science, Mathematics, etc). The norm can then be expressed as GPA = 5. We assume that the threshold associated with this norm is 0.2, meaning that in some special cases a graduate student candidate with a minimum of 4.8 can be admitted as well. The threshold can be expressed as follows: GPA > 4.8

For the faculty member (FM) the norm is to ensure team cohesion. This rather abstract norm can be lead to two more concrete norms: FM_Norm 1 = *"motivated graduate student candidate" and* FM_Norm 2 = *"*Graduate *student candidate with relevant skill set."*

For the FM, we define two variables that correspond to this norm: the graduate student candidate's motivation for a particular subject defined by the FM and the skills developed in the relevant area. The thresholds can be expressed as follows: "the graduate student candidate expresses interest in the subject (e.g. in her motivation letter or interview)"; "the graduate student candidate has at least 2 years of experience working in the area."

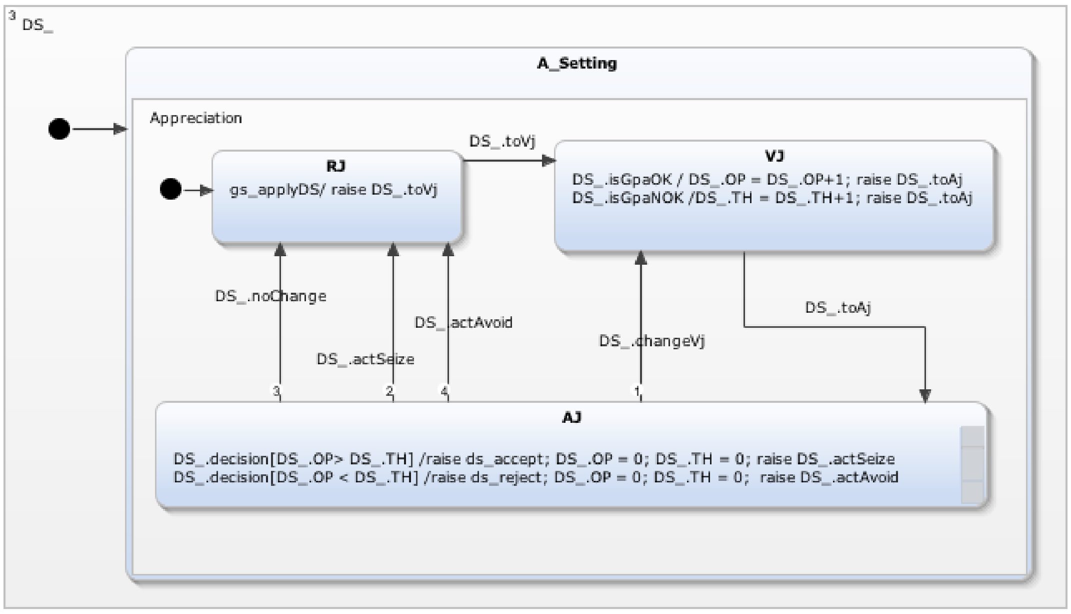
In the next sections, we explain this transformation of observations into responses in detail for the DS and FM.

## 5.2 Statechart Model of the DS Appreciative System

The statechart model of the DS appreciative system is shown in Figure 9.

### Step 1: DS's Reality Judgment

In the RJ state, we define an event (an observation) that triggers the regulating process. For the DS, it is the gs_applyDS event raised in the operational model (Figure 13). This corresponds to the GS application for the doctoral school. Once this event is captured in the RJ (i.e. a graduate student candidate application is received), the relative reality judgments are created (analogy of ri in Section 4.2). In the case of DS, this corresponds to the graduate student candidate's GPA (e.g. based on her track of academic records).



3 DS_
A_Setting
Appreciation
RJ
gs_applyDS/ raise DS_.toVj
DS_.toVj
VJ
DS_.isGpaOK / DS_.OP = DS_.OP+1; raise DS_.toAj
DS_.isGpaNOK /DS_.TH = DS_.TH+1; raise DS_.toAj
DS_.noChange
DS_.actAvoid
DS_.actSeize
DS_.changeVj
DS_.toAj
3
2
4
1
AJ
DS_.decision[DS_.OP> DS_.TH] /raise ds_accept; DS_.OP = 0; DS_.TH = 0; raise DS_.actSeize
DS_.decision[DS_.OP < DS_.TH] /raise ds_reject; DS_.OP = 0; DS_.TH = 0;  raise DS_.actAvoid

**Figure 9.** DS_ – the Appreciation statechart for the DS

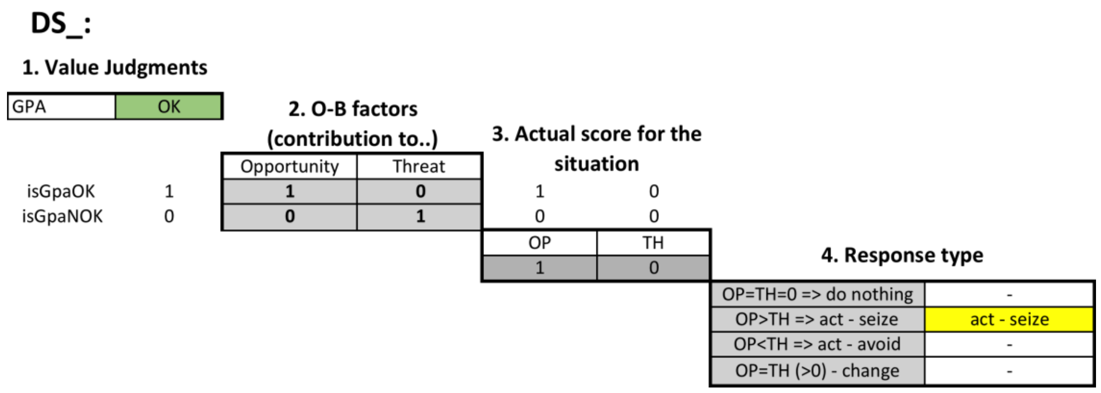
### Step 2: DS's Value Judgment

We model two events that correspond to the positive and negative evaluation of the graduate student candidate with respect to the threshold specified above: isGpaOK is raised when the graduate student candidate's GPA is within the threshold (>4.8) and isGpaNOK is raised when this threshold is crossed.

### Optimizing-balancing

In order to understand how value judgments will be transformed into a response, it is important to refer to the context of the PhD recruitment process. We model this process based on the assumption that the DS is searching for graduate student candidates. In other terms, there exists a norm related to a (desired) number of PhD students defined by the DS, and, regarding this norm, we are below some threshold (ex: #PhD > *x*). This is not shown in the model but embedded into the context – this is the reason why the university runs the PhD recruitment process and the DS participates in this process on the first place.

Figure 10 illustrates how the evaluation of a graduate student candidate's application (i.e. GPA) leads to a decision by the DS (i.e. optimizing-balancing):



**DS_:**

**1. Value Judgments**

| GPA | OK |
|---|---|

**2. O-B factors (contribution to..)**

| | | Opportunity | Threat |
|---|---|---|---|
| isGpaOK | 1 | 1 | 0 |
| isGpaNOK | 0 | 0 | 1 |

**3. Actual score for the situation**

| | |
|---|---|
| 1 | 0 |
| 0 | 0 |
| OP | TH |
| 1 | 0 |

**4. Response type**

| | |
|---|---|
| OP=TH=0 => do nothing | - |
| OP>TH => act - seize | act - seize |
| OP<TH => act - avoid | - |
| OP=TH (>0) - change | - |

**Figure 10.** Optimizing-balancing for the appreciation statechart of the DS

Here the graduate student candidates with GPA = 4.8 and higher are considered as an opportunity for the doctoral school to maintain the norm regarding the desired number of PhD students. Whereas the graduate student candidates with GPA < 4.8 represent a threat as, if recruited, they might fail to complete the PhD program. In the statechart in Figure 10 this optimizing-balancing is specified with the corresponding statements:

```
DS_.isGpaOK / DS_.OP = DS_.OP +1; raise DS_.toAj
DS_.isGpaNOK /DS_.TH = DS_.TH +1; raise DS_.toAj
```

### Step 3: DS's Action Judgment

In AJ, we define the responses that will be triggered based on the optimizing-balancing above:

```
DS_.decision[DS_.OP> DS_.TH] /raise ds_accept; DS_.OP = 0; DS_.TH = 0; raise
DS_.actSeize
DS_.decision[DS_.OP < DS_.TH] /raise ds_reject; DS_.OP = 0; DS_.TH = 0;  raise
DS_.actAvoid
```

In case of opportunity, the graduate student candidate will be accepted (raise ds_accept); and in case of threat the graduate student candidate will be rejected (raise ds_reject). Other response types (i.e. noChange or changeVj) are not defined for this model, since the optimizing and balancing is simple and based on one value judgment only.

## 5.3 Statechart Model of FM Appreciative System

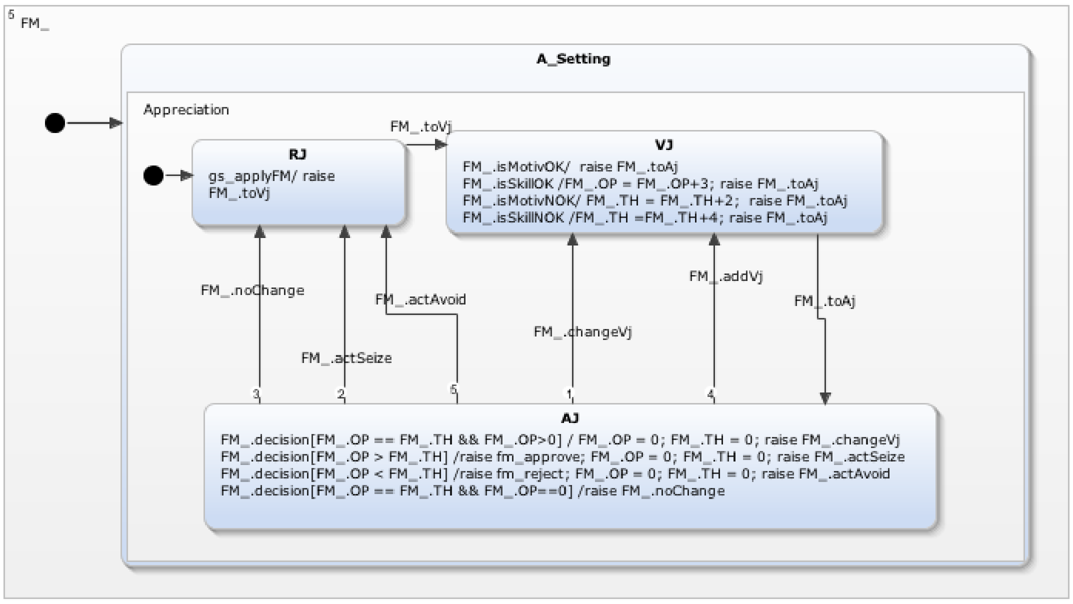
The appreciative system of FM is presented in Figure 11.

### Step 1: FM's Reality Judgment

In the RJ state, we define an event (an observation) that triggers the regulating process. For the FM, it is the gs_applyFM event raised in the operational model (Figure 10, Figure 11). This corresponds to the GS application sent directly to the faculty member. Once this event is captured in the RJ, the relative reality judgments are created (analogy of ri in Section 4). In the case of FM, this corresponds to the graduate student candidate's Motivation (based on her motivation letter) and Skill (based on her CV).

## Step 2: FM's Value Judgment

We model four events that correspond to the evaluation results of these variables: isMotivOK is raised when the graduate student candidate does express the interest to the subject in her motivation letter (see Section 5.1) and isMotivNOK in the opposite case. Idem, isSkillOK is raised if the graduate student candidate's CV shows 2 or more years of experience in the relevant area and isSkillNOK is raised otherwise.



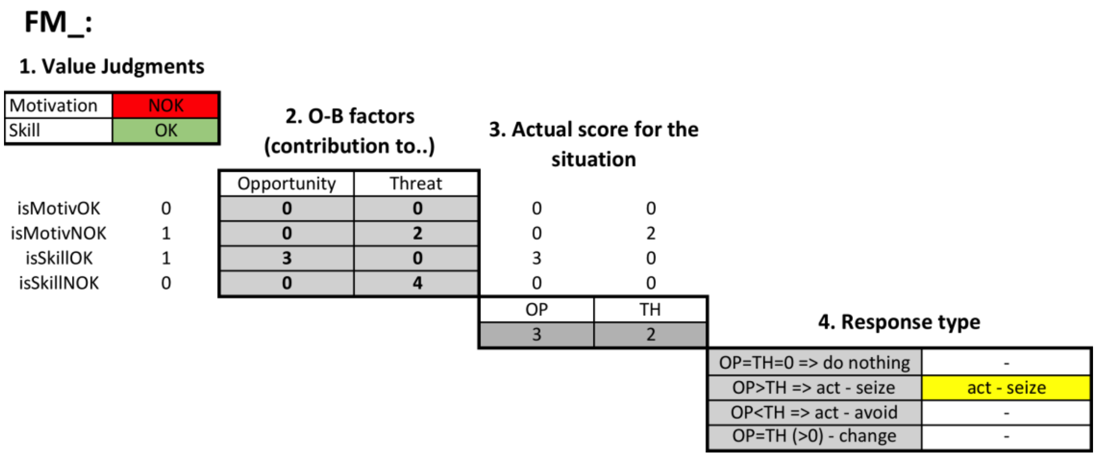
5 FM_
A_Setting
Appreciation
FM_.toVj
RJ
gs_applyFM/ raise FM_.toVj
VJ
FM_.isMotivOK/ raise FM_.toAj
FM_.isSkillOK /FM_.OP = FM_.OP+3; raise FM_.toAj
FM_.isMotivNOK/ FM_.TH = FM_.TH+2; raise FM_.toAj
FM_.isSkillNOK /FM_.TH =FM_.TH+4; raise FM_.toAj
FM_.noChange
FM_.actAvoid
FM_.actSeize
FM_.changeVj
FM_.addVj
FM_.toAj
3
2
5
1
4
AJ
FM_.decision[FM_.OP == FM_.TH && FM_.OP>0] / FM_.OP = 0; FM_.TH = 0; raise FM_.changeVj
FM_.decision[FM_.OP > FM_.TH] /raise fm_approve; FM_.OP = 0; FM_.TH = 0; raise FM_.actSeize
FM_.decision[FM_.OP < FM_.TH] /raise fm_reject; FM_.OP = 0; FM_.TH = 0; raise FM_.actAvoid
FM_.decision[FM_.OP == FM_.TH && FM_.OP==0] /raise FM_.noChange

**Figure 11.** FM – the appreciation statechart for the FM

### Optimizing-balancing

As above, it is important to refer to the context of the PhD recruitment process. We make an assumption that the faculty member (FM) is interested to recruit a PhD student in her team. Thus, there exists a norm related to a (desired) number of PhD students in the team defined by the FM, and, regarding this norm, we are below some threshold (ex: #PhDTeam > y*)*. This is not shown in the model but embedded into the context and this is the reason why the FM is participating in the PhD recruitment process on the first place.

Figure 12 illustrates how the evaluation of a graduate student candidate's application (i.e. Skill and Motivation) leads to a response by the FM (i.e. optimizing-balancing):



**FM_:**

**1. Value Judgments**

| | |
|---|---|
| Motivation | NOK |
| Skill | OK |

**2. O-B factors (contribution to..)** / **3. Actual score for the situation**

| | | Opportunity | Threat | | |
|---|---|---|---|---|---|
| isMotivOK | 0 | 0 | 0 | 0 | 0 |
| isMotivNOK | 1 | 0 | 2 | 0 | 2 |
| isSkillOK | 1 | 3 | 0 | 3 | 0 |
| isSkillNOK | 0 | 0 | 4 | 0 | 0 |

| OP | TH |
|---|---|
| 3 | 2 |

**4. Response type**

| | |
|---|---|
| OP=TH=0 => do nothing | - |
| OP>TH => act - seize | act - seize |
| OP<TH => act - avoid | - |
| OP=TH (>0) - change | - |

**Figure 12.** Optimizing – balancing for the appreciation statechart of the FM

We model a FM who gives a clear priority to the graduate student candidate skills: isSkillOK contributes 3 for the opportunity and isSkillNOK contributes 4 to threat. Motivation, in contrast, is necessary but not sufficient condition for the graduate student candidate to be approved: a good motivation alone does represent neither threat nor opportunity for the FM. This is represented by an o-b factor for isMotivOK = [0, 0]; and the o-b factor for isMotivNOK = [0, 2].

In the statechart in Figure 11 this optimizing-balancing is specified with the corresponding statements:

```
FM_.isMotivOK/  raise FM_.toAj
FM_.isMotivNOK/ FM_.TH = FM_.TH +2;  raise FM_.toAj
FM_.isSkillOK /FM_.OP = FM_.OP +3; raise FM_.toAj
FM_.isSkillNOK /FM_.TH = FM_.TH +4; raise FM_.toAj
```

The actor may choose to evaluate two variables or only one prior to define a response.

Example: the FM receives the application and evaluates the motivation first. If motivation is not convincing (isMotivNOK) – the optimizing-balancing will terminate with OP = 0, TH = 2 and the response will be to avoid the threat. However, if the FM proceeds with evaluation of skills (i.e. adds another value judgment) and finds them satisfactory (isSkillOK) – the optimizing-balancing result will be: OP = 3; TH = 2 and the response will be to seize the opportunity.

The optimizing-balancing factors affect the way decisions are made. Further details on this subject, however, are out of the scope of this article.

### Step 3: FM's Action Judgment

In AJ, we define the responses that will be triggered based on the optimizing-balancing above:

```
FM_.decision[FM_.OP == FM_.TH && FM_.OP>0] / FM_.OP = 0; FM_.TH = 0; raise
FM_.changeVj
FM_.decision[FM_.OP > FM_.TH] /raise fm_approve; FM_.OP = 0; FM_.TH = 0; raise
FM_.actSeize
FM_.decision[FM_.OP < FM_.TH] /raise fm_reject; FM_.OP = 0; FM_.TH = 0; raise
FM_.actAvoid
FM_.decision[FM_.OP == FM_.TH && FM_.OP==0] /raise FM_.noChange
```

In case of opportunity, the graduate student candidate will be approved by the FM (raise fm_approve); and in case of threat the graduate student candidate will be rejected (raise fm_reject).

The example above shows that the decision depends on the valuation process. We find it important to highlight the complexity of the valuation process and its subjectiveness since it is related to the social nature of the appreciation as a whole. We will study this process in our future work.

## 5.4 Appreciative and Operational Models Working Together

Figure 13 illustrates the three actors' operational model and appreciative model of the PhD recruitment process. The operational model can be considered as the technical perspective and the appreciative model as the social perspective.

The appreciative models illustrate how the decisions regarding accepting, approving and rejecting graduate student candidates are taken and how they are related to the social norm held by the corresponding process actors.

Extending the operational model with the appreciative model allows one to separate the decision-making *process* from the decision *results* observed in the process and to reason about the process efficiency.

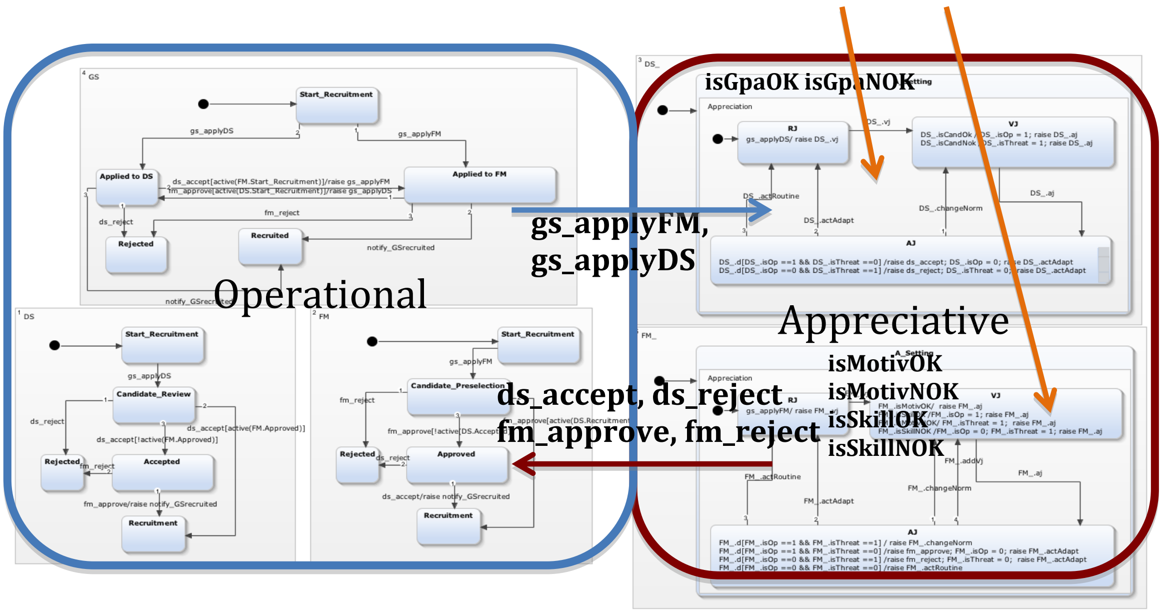


**Figure 13.** Extended model of the PhD recruitment process. Interactions between the operational model and the appreciative model are illustrated with bold colored arrows showing the corresponding "in" and "out" events.

## 5.5 Model Simulation

Simulating the extended model in YAKINDU allows for visualizing the process scenarios as in the operational process model. As above, some of these events need to be raised by the user, using the simulation interface, and some are automatically raised by the statechart.

The DS_ and FM_ statecharts (representing the social perspective in Figure 13) capture the two events raised in the operational process: gs_applyFM and gs_applyDS. These events trigger the regulative processes at FM_ and DS_ respectively as explained in Sections 5.2 and 5.3. During the regulative process, DS_ and FM_ make value judgments isGpaOK/NOK, isSkillOK/NOK, isMotivOK/NOK respectively.

In the operational perspective DS, FM and GS statecharts capture four events raised by the DS_ and FM_ in response: ds_accept, ds_reject, fm_approve, fm_reject. These events affect the formal process the same way as in the models in Figure 1, Figure 2 and Figure 3.

Figure 15 – (a) illustrates the configuration of the statechart that corresponds to the configuration in Figure 5 (a) – after the user triggers gs_applyDS event from the simulation window. The configurations of the DS, FM and GS statecharts that correspond to the operating model of the PhD recruitment process do not change.

The FM_ statechart (bottom-right) is in the starting state (RJ). The DS_ statechart (top-right) is in the VJ state: gs-applyDS triggered this value judgment.

From this configuration, the next step can be to "simulate" the appreciative system of DS (DS_). To do so, the user needs to "evaluate" the graduate student candidate application and to raise isCandOK or isCandNOK event from the DS_ interface of the simulator window (Figure 15 (b)).

```
interface DS_:
            in event toVj                 //-automatic, raised when RJ is created
in event toAj                             //-automatic, raised when VJ is made
            in event addVj                //-raised by the user to make a new VJ
            in event decision             //-raised by the user when decision making is finished
// response types  - automatic, based on optimizing-balancing
in event noChange
            in event actSeize
            in event actAvoid
            in event changeVj
//value judgments           - manual, raised by the user
            in event isGpaOK
            in event isGpaNOK
//Optimizing-balancing
            var OP: integer
            var TH: integer

 interface FM_:
            in event toVj                 //-automatic, raised when RJ is created
in event toAj                             //-automatic, raised when VJ is made
            in event addVj                //-raised by the user to make a new VJ
            in event decision             //-raised by the user when decision making is finished
// response types  - automatic, based on optimizing-balancing
in event noChange
            in event actSeize
            in event actAvoid
            in event changeVj
//value judgments           - manual, raised by the user
            in event isMotivOK
            in event isMotivNOK
            in event isSkillOK
            in event isSkillNOK
//Optimizing-balancing      - set up automatically
            var OP: integer
            var TH: integer

internal:
//Process events: interaction between the Operational and Social process perspectives
            event gs_applyDS          //- From Operational (GS) to Social (DS_); raised by the user
            event gs_applyFM          //- From Operational (GS) to Social (FM_); raised by the user
            event fm_approve          //- From Social (FM_) to Operational; (raised automatically)
            event fm_reject           //- From Social (FM_) to Operational; (raised automatically)
            event ds_accept           //- From Social (DS_) to Operational; (raised automatically)
            event ds_reject           //- From Social (DS_) to Operational; (raised automatically)
            event notify_GSrecruited  //- raised automatically
```

**Figure 14.** The list of events from the extended process model. These events represent the actual judgments made in the mind of the actor. During the simulation, the corresponding events have to be triggered manually.

The decision about accepting or rejecting the graduate student candidate will be selected automatically by the simulator from the list of statements defined in the AJ state of DS_.

Once the DS_ statechart is in the AJ state, the user can raise the "decision" event from the DS_ interface of the simulator window. The decision will be executed automatically based on the optimizing and balancing result, as explained in the previous section. Thus, ds_accept or ds_reject events will be raised automatically. The statecharts representing the operational process will react upon it accordingly: in the first case, gs_applyFM will be raised and the evaluation of the graduate student candidate by FM will be simulated; in the second case the graduate student candidate will be rejected and the process will stop.

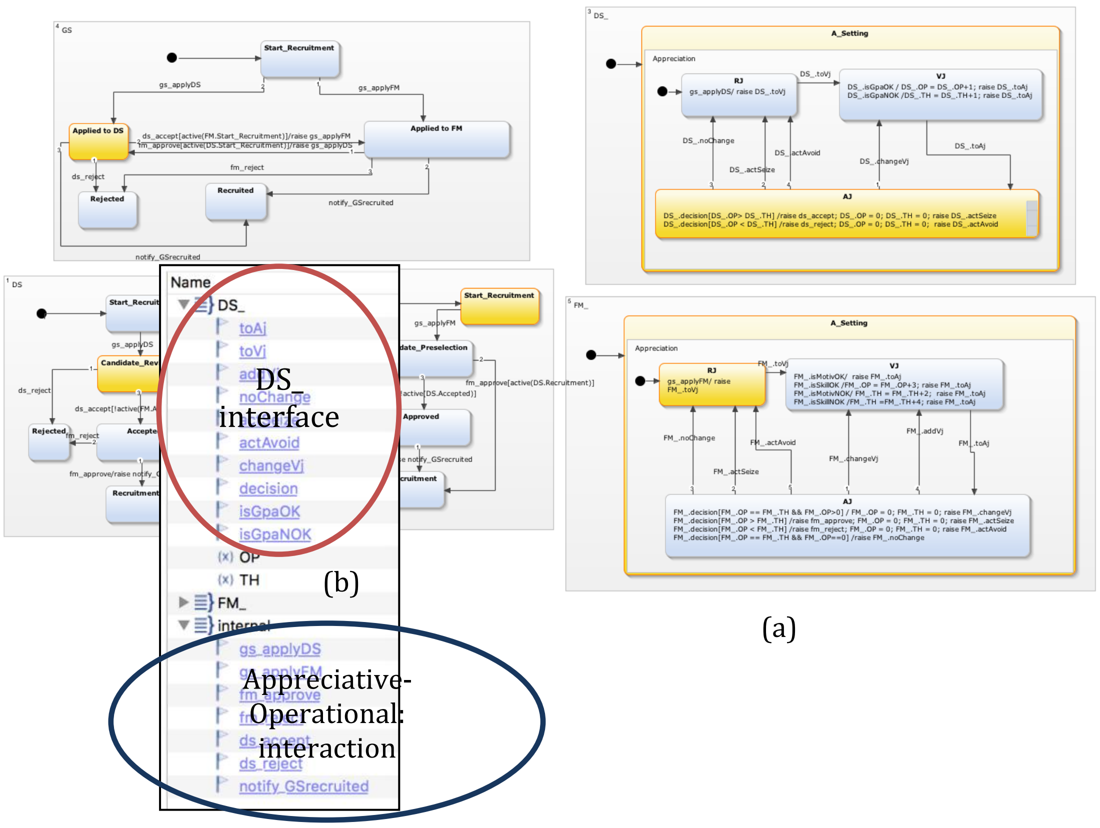


**Figure 15.** Simulation of the extended process model with YAKINDU

### 5.6 PhD Recruitment Efficiency: Discussion – Explaining the Gap

The VJ states of the appreciative systems of FM and DS show that the overall decision depends on the evaluation of three variables: graduate student candidate background (by DS), her motivation and skills (by the FM).

The PhD recruitment process efficiency is related to the capacity to provide equal opportunities for the university professors to recruit doctoral students. Thus, technically, all FM should have the same chances to recruit a PhD student within this process. Considering this extended model, we can see that this is only possible if the variables measured in VJ states by DS and by FM are **independent**.

The fact that some professors cannot recruit the PhD students through this process can be explained by existing (implicit) dependencies between these variables. We illustrate these dependencies with the following examples:

**Example 1:** *The student with a strong technical background often finds the subjects offered by the FM "too soft."* In the model we would observe this as follows: the evaluation isGpaOK made by DS in the VJ state of DS_ will often correlate during the simulation with the evaluation isMotivNOK made by FM in the VJ state of FM_ resulting in TH=2 for a given graduate student candidate. (Unless the skills of the candidate are good – she will not be approved by the FM.) Thus, the FM will rarely approve the graduate student candidates accepted by the DS.

**Example 2**: *The students with a strong technical background most of the time have insufficient skills for the subject as defined by the FM (i.e. the subject requires skills in social or management sciences).* In the model we would observe this as follows: the evaluation isGpaOK made by DS in the VJ state of DS_ will often correlate with the evaluation isSkillNOK made by FM in the VJ state (TH=3) for a given graduate student candidate. Thus, the FM will rarely approve the graduate student candidates accepted by DS.

**Example 3:** *The student with a strong motivation and skills for the subject often has insufficient technical background as defined by the DS.* In the model we would observe this as follows: the evaluation isSkillOK and isMotivOK made by FM in the VJ state of FM_ will often correlate with the evaluation isGpaNOK made by DS in the VJ state of DS_ for a given graduate student candidate. Thus the graduate student candidates approved by the FM will be often rejected by the DS.

The examples above describe the specific situations that are nevertheless common for professors working on transversal subjects. They show that **the norms the process participants hold do not fit all participants**.

DS_Norm =“*To ensure strong background of a graduate student candidate in the research areas defined by the school*”

FM_Norm =*“To ensure team cohesion through recruiting motivated graduate student candidates with relevant skill set”*

Ensuring strong background may run contrary to the cohesion of the FM team (while possibly increasing the background of all graduate students). We can observe here the balancing act between the interest of the individual (i.e. the faculty member) and the group (i.e. the doctoral school). This is the essence of social systems, creating and preserving social groups through social norms.

## 6 Discussion on the Formalization of Social Systems

In this article we present how we can model the social perspective and the technical perspective of a business process. One challenge with the social perspective is that humans or living beings, can adapt and have intuitions. These are very challenging to formalize, and probably – luckily – yet impossible.

The model presented in this article can be used to simulate predefined scenarios. We model appreciative settings that do not change and we rely on an environment that is stable. Few thoughts on the limitations of our approach are as follows:

**On reality judgments:** We do not discuss how the actor perceives the environment (reality judgment), uses her intuition and how she imagines disrupting the environment (value judgment and action judgment). This is out of the scope for our current work but is the key to model more sophisticated ways to adapt.

**On value judgments:** We model the regulative process where the actor will need to change her appreciative setting (either reality judgment, or value judgment or optimizing-balancing factors). During the simulation, we demonstrate that the actor can decide to add a new value judgment or to change the existing one (“change her mind”). However, the current model does not explain how appreciative setting can be changed. In the current version, changing of optimizing-balancing factors and reality judgments requires model redesign and cannot be done at run time.

**On action judgments:** In any social group, there will be numerous relationships between appreciative systems and their underlying norms. In a similar way, the choice of our response (in action judgment) may require an interaction with more than one operational process. Another appreciative system can model this choice. Different operational processes can be chosen or constructed following the response strategy. The response strategy (e.g. act and seize or act and avoid) is based on the actor’s understanding and anticipation of the environment. However, we do not show how the actor is gaining this knowledge.

**On formalizing social behavior:** The model we have shown is a mechanistic simplification of a much more sophisticated way in which social groups behave. Still, it allows rationalizing our behavior instead of just ignoring all social and psychological aspects.

**On statecharts and YAKINDU:** state-transition paradigm to reason about both technical and social perspectives of a business process. This paradigm is well known for modeling systems behavior, whereas process models most commonly follow the activity-oriented paradigm, representing a process as a predetermined sequence of activities. Within the state-oriented paradigm, the process scenario is specified with as a sequence of events; the concrete activities that will produce these events remain implicit. Examples of state-oriented modeling formalisms include state machines in UML [18], generic state-transition systems or state machines, such as FSM [7] or Petri Nets [19], and statecharts by D. Harel [6] created for the specification and analysis of complex discrete-event systems. Harel's statechart and the YAKINDU statecharts modeling tool are proposed for modeling scenarios for decision making support for knowledge-intensive, weakly structured processes in [20], [21].

## 7 Conclusion

In this article we show how to extend business process modeling by using appreciative systems. We define an operational model (the statechart equivalent of a BPMN model) coupled with an appreciative model (statechart equivalent of the mindset of participants). Using the appreciative system in the context of BPMN has triple interest:

1. For process engineers: we take into account not only a process and its context, but a social system formed by the process participants and their (personal and professional) contexts. This gives an opportunity to discover more complex scenarios and to anticipate the sources of actions with undesirable (and for some processes, catastrophic) consequences. Safety critical processes, activities associated with risk taking, can benefit from this socio-technical approach for process modeling and analysis.
2. For process participants: the appreciative model provides an opportunity to think about what people perceive (reality judgment) their values (value judgment) and their response strategy (action judgment). Identifying these elements, the process participants can better cope with the stress encountered during their activities.
3. For process managers: understanding of (or at least being aware of) the appreciative systems of process participants can help the process managers to anticipate and understand the source of conflicting situations. The process manager can be better equipped to propose a conflict resolution strategy.

We have specifically shown that the social perspective can explain the problems related to the operational processes in the organizations and, potentially, can help anticipate issues due to conflicts of norms, and to resolve observed problems by showing elements to take into considerations when redesigning these operational processes.

Our work can be placed in the mid-way between the social sciences and engineering. In the social sciences, the problems of cognition, decision-making and social interactions of an individual are addressed in a much more elaborated manner than we attempt in this article. Engineering, in contrast, tends to address the same issues in a much more pragmatic and, sometimes, oversimplified manner. Our goal is to find an appropriate level of complexity and to bring the values of the social sciences to engineering.

This article illustrates how actors can maintain their stability with an appreciative setting that does not evolve (e.g. no new judgments). The goal of the appreciative model is mostly pedagogical, nevertheless we assume that it brings value to process engineers, process participants and managers by making them aware of the mechanisms at stake.

## References

[1] S. G. Vickers, *Value Systems and Social Process*, Tavistock Publications, London, 1968.

[2] S. G. Vickers, "Policymaking, Communication, and Social Learning," Adams, G.B., Forester, J., Catron, B.L., (Eds): *Transaction Books*. New Brunswick NJ, 1987.

[3] S. G. Vickers, *The Art of Judgment: A Study of Policy Making*, Sage, 1968.

[4] W. B. Cannon, “Organization for Physiological Homeostasis,” *Physiological Reviews*, vol. IX, no. 3, pp. 399–431, 1929. Available: https://doi.org/10.1152/physrev.1929.9.3.399

[5] I. Rychkova, G. Regev, “From Goal-Achievement to the Maintenance of Relationships: Extending Business Process Models with Homeostasis and Appreciation,” *Proceedings of 4th International Workshop on Socio-Technical Perspective in IS development (STPIS'18)*, CEUR, vol. 2107, pp. 87–100, 2018.

[6] D. Harel, “Statecharts: A visual formalism for complex systems,” *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987. Available: https://doi.org/10.1016/0167-6423(87)90035-9

[7] A. Gill, “Introduction to the theory of finite-state machines,” 1962. Available: https://archive.org/details/IntroductionToTheTheoryOfFiniteStateMachines

[8] YAKINDU STC. Available: www.itemis.com/en/yakindu/state-machine/

[9] A. Rosenblueth, N. Wiener, and J. Bigelow, “Behavior, Purpose and Teleology,” *Philosophy of Science*, vol. 10, no. 1, pp. 18–24, 1943. Available: https://doi.org/10.1086/286788

[10] S. J. Cooper, “From Claude Bernard to Walter Cannon. Emergence of the concept of homeostasis,” *Appetite*, vol. 51, no. 3, pp. 419–427, 2008. Available: https://doi.org/10.1016/j.appet.2008.06.005

[11] G. Regev, O. Hayard, and A. Wegmann, “What We Can Learn About Business Modeling From Homeostasis,” *Business Modeling and Software Design. BMSD 2012. LNBIP*, vol. 142, pp. 1–15, Springer, 2013. Available: https://doi.org/10.1007/978-3-642-37478-4_1

[12] G. M. Weinberg and D. Weinberg, *General Principles of Systems Design*, Dorset House, 1988.

[13] G. Regev and A. Wegmann, “Where do Goals Come From: the Underlying Principles of Goal-Oriented Requirements Engineering,” *Proceedings of the 13th IEEE International Requirements Engineering Conference (RE’05)*, 2005. Available: https://doi.org/10.1109/RE.2005.80

[14] G. Regev and A. Wegmann, “Revisiting Goal-Oriented Requirements Engineering with a Regulation View,” *Business Modeling and Software Design. BMSD 2011. LNBIP*, vol. 109, pp. 56–69, Springer, 2012. Available: https://doi.org/10.1007/978-3-642-29788-5_4

[15] A. Dardenne, A. van Lamsweerde, and S. Fickas, “Goal Directed Requirements Acquisition,” *Science of Computer Programming*, vol. 20, no. 1–2, pp. 3–50, 1993. Available: https://doi.org/10.1016/0167-6423(93)90021-G

[16] A. I. Anton and C. Potts, “The use of goals to surface requirements for evolving systems,” *Proceedings of the 20th International Conference on Software Engineering*, 1998. Available: https://doi.org/10.1109/ICSE.1998.671112

[17] G. Regev, O. Hayard, and A. Wegmann, “Service Systems and Value Modeling from an Appreciative System Perspective,” *Exploring Services Science. IESS 2011. LNBIP*, vol. 82, pp. 146–157, Springer, 2011. Available: https://doi.org/10.1007/978-3-642-21547-6_12

[18] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified Modeling Language Reference Manual*, The (2Nd Edition). Pearson Higher Education, 2004.

[19] T. Murata, “Petri nets: Properties, analysis and applications,” *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989. Available: https://doi.org/10.1109/5.24143

[20] E. Kushnareva, I. Rychkova, R. Deneckére, and B. Le Grand, “Modeling crisis management process from goals to scenarios,” *Business Process Management Workshops. BPM 2016. LNBIP*, vol. 256, pp. 55–64. Springer, 2016. Available: https://doi.org/10.1007/978-3-319-42887-1_5

[21] I. Rychkova, B. Le Grand, and C. Souveyet, “Towards Executable Specifications for Case Management Processes,” *Advances in Intelligent Process-Aware Information Systems*, vol. 123, pp. 49–77, Springer, 2017. Available: https://doi.org/10.1007/978-3-319-52181-7_3