Stability of Self-Adaptive Software: A Case Study

Mieczyslaw M. Kokar[1], Kenneth Baclawski[1]
, Yong Xun[1], and Kevin M. Passino[2]

[1] Northeastern University, Boston, Massachusetts, USA
kokar@coe.neu.edu
kenb@ccs.neu.edu
[2] Ohio State University, Columbus, Ohio, USA,
passino@ee.eng.ohio-state.edu

Abstract. The paradigm of self-controlling software is becoming recognized as a way of achieving both better performance and robustness in the presence of unexpected changes in the environment. However, self-control (often referred to as *self-adaptation*), if not properly designed, can lead to some emerging behaviors that may not only lower the performance of the system, but even lead to catastrophic outcomes. In this paper we show an example of a self-controlling system based on the control metaphore. The system is a dynamic resource allocator (scheduler). We first show a model of such a system. Then we simulate its behavior and show that the self-controlling function can lead to unstable behavior of the whole system. We discuss ways to deal with such a problem.

1 Introduction

2 An Example of a Self-Controlling System

In this paper we describe a resource management system whose main part is a scheduler. The goal of the scheduler it to allocate a receiver (resource) to a particular radar (task). It is assumed that there are n radars in the environment. Each of the radars rotates continuously. However, it illuminates only in the direction it is pointing to at a particular time. There is also one controller for each radar. The contoller computes the desired time of the next dwell on the radar. Both the controller and the scheduler need a computer to perform their operations. Therefore, the computer class is part of the system model. Figure 1 represents the UML class diagram of the whole system.

We model radars as timed automata. We assume that each radar's beam width is 2^o. Consequently, the radar can point in 180 different directions. We model this with 180 states, one for each direction. It points in a direction for the time interval dt and then switches to the next state. The direction depends on the $X - Y$ coordinates of the radar, in addition to the state it is in. For the sake of interaction with the Receiver we defined an interaction variable $radar - i$, for each radar. The value of this variable keeps track of the direction that the radar

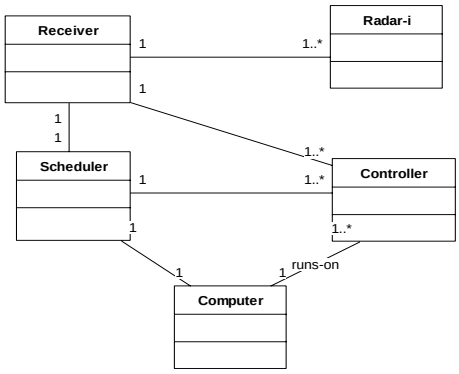

Fig. 1. The Resource Management System

is pointing to. This variable contains the equation of the line and the direction. It is represented as a triple $< a, b, d >$, where a and b are the coefficients of the line equation $ax + b$ and d is the radar's pointing direction i.e., the angle with respect to the $West - East$ line. The state transition diagram for a radar (in this

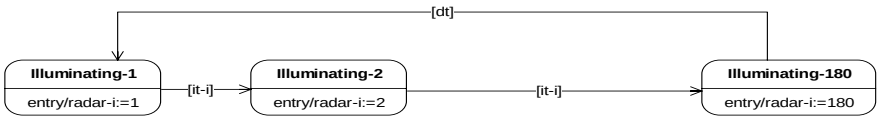

Fig. 2. The behavior of the Radars

The receiver is controlled by the scheduler (see Figure 4). The scheduler generates schedulig events ($sched - 1, \ldots, sched - n$). The receiver, in response to an event $sched{-}i$ switches to the state $PointingAtRadar{-}i$ for the duration of the dwell time prescribed by the scheduler. When pointing at a particular radar, the receiver either detects the radar or not, depending on whether the receiver is currently located on the direction in which the radar is illuminating during this dwell time or not (see Figure 3). If the receiver is on that line, the receiver

generates an event of detection, i.e., $detect - i$ is set to 1. Otherwise, $detect - i$ is set to 0. Since the receiver is on a moving platform, the line with respect to a particular radar changes constantly. Thus even if the receiver is on the line of a particular radar at the beginning of its dwell, it may be outside of the line at some point in the middle of the dwell. Consequently, if this happens, the radar cannot be detected. For this reason, the $detect - i$ event must be generated only when the receiver is within the line of view of the radar for the whole dwell period.

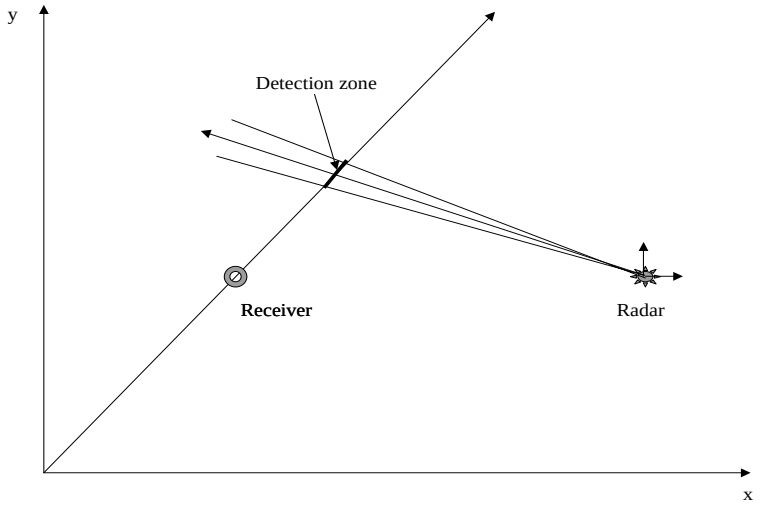

Fig. 3. The detection scenario

The receiver is modeled as a timed automaton as in Figure 4. Its continuous dynamics is given by the equation

$$\frac{dV}{dt} = 0 \tag{1}$$

where $V(0) = V_0$. In other words, the receiver is moving with a constant velocity V_0.

The detection event is passed to an appropriate controller. In this system there is one controller for each radar. In response to a detection (or non-detection) event the controller needs to compute the time at which the receiver should be pointed again at that radar (next dwell). To perform this action the controller

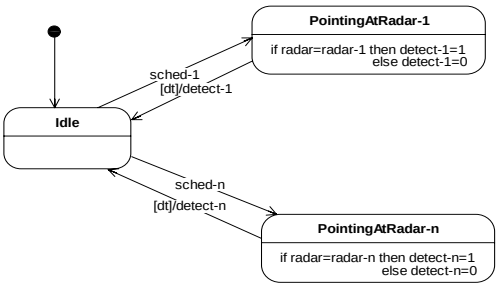

Fig. 4. The behavior of the Receiver

needs the computer. Consequently, it first sends the *compute* event to the computer and then invokes its control law to compute the time of next dwell on that radar. After this is done, the controller sends the next dwell time to the scheduler and then releases the computer by sending the *release* event. The state transition diagram for the controller is shown in Figure 5.

After the scheduler receives the $dwell - i$ event it re-computes its schedule. A schedule consists of a list of Control Descriptor Words (CDWs). Each CDW contains information on the radar id, the start time of next and the duration of the dwell. The CDW event is then sent to the receiver. Similarly as with the controller, the scheduler needs the computer to perform its computation. And consequently, it sends a *compute* event to the computer at the start of computation, and then the *release* event when the computation is complete. The state transition diagram for the scheduler is shown in Figure 6.

Finally, we discuss the last system component - the computer. Since this system is a self-aware system, the computer needs to be represented explicitly. The computer is shown to be invoked by the controller and by the scheduler by sending the *compute* event. It is released by the *release* event received from either the controller or the scheduler. If the computer is computing the next control input, and thus is in the $ComputingControl$, then it can be released only by the controller; similarly for the scheduler. The state transition diagram for the computer is shown in Figure 7.

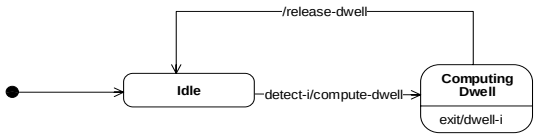

Fig. 5. The Controller

3 Stability

The performance of this system is measured in terms of probability of detection (P_d) defined as the number of detections of a particular radar over the number of possible detections (the number of rotations of the radar). The goal of the scheduler is to direct the receiver towards the radars so that (to dwell on the radar) at exactly the time that the radar's illumination line points towards the receiver. In our system the scheduler executes the Earliest Deadline First (EDF) scheduling policy. The outcome of this policy, however, depends on the output of the controller, which tells the radar when this coincidence is expected, i.e., the time when the radar is illuminating in the direction of the receiver.

The system is stable (intuitively) if none of the radars is ignored for some pre-specified time interval B_t. There are many definitions of stability in the control literature. In this paper we use the *stability in the sense of Lagrange.*

A system is said to be "stable in the sense of Lagrange" [**?**] if for every initial condition the $P_{di}(0)$) such that the initial condition lies within a certain bound (e.g., $\sum_{i=1}^{N} P_{di}(0) \leq \alpha$) there exists a bound B such that $P_d \leq B$ is satisfied (note that B may depend on α). Clearly, stability in the sense of Lagrange is simply a type of boundedness property. Typically, in practical applications, we would like B to be as small as possible.

Lagrange stability only says that *there exists* a bound for the uncertainty trajectories. A slightly stronger stability condition is that of *uniform ultimate boundedness* (UUB), where for every initial condition the trajectories are bounded, and as time goes to infinity, they will all approach a B-neighborhood of the origin where we know the bound B (e.g., it typically depends on the parameters of the problem.

Note that if the scheduler has chosen a specific i (specific radar) for a long enough time it may be possible to reduce P_{di} to near one. When this happens it normally does not make sense to keep the receiver focused on that radar

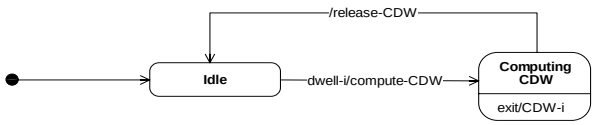

Fig. 6. The Scheduler

(however, if it is a very high priority radar then we may want to maintain as much information about it as possible). It is clear that to be able to achieve boundedness, the scheduler cannot ignore any one radar for too long (i.e., the "revisit time" for any one target cannot be too long) or its corresponding P_{di} will rise to a high value.

In this paper we present simulations of various scenaria. An example of one such simulation is shown in Figure 8. This figure shows the performance of the scheduling system when the gain of the controller (K_p) is set to 0.1. This figure shows plots of two parameters: miss ratio (P_{mi}) for five radars, and probability of detection (P_{di}) the same these targets. The probability of detection is the performance measure (QoS) of the whole system. This measure, as we stated earlier in the paper, depends on a number of factors: the rotations of the radars, the movent of the receiver platform, the decisions made by the scheduler and the decisions made by the controller. Of particular interest are the decisions made by the scheduler. However, since these decisions depend on the inputs from the controller, the resultant performace depends on both the scheduler in the controller. For this reason we showed the miss ratio in the left column of Figure 8 and the probability of detection in the right column of this figure. The miss ratio tells us what percentage of the controller's decisions were ignored by the scheduling policy. In other words, even though the controller tells the scheduler to schedule a specific radar at a specific time, the scheduler, due to the heavy load (saturation) of the receiver, misses some of its requests.

4 Conclusions

As can be seen from Figure 8, the system is unstable. Only for targets 2 and 4 the probability of detection is stable. For the other three targets the performance

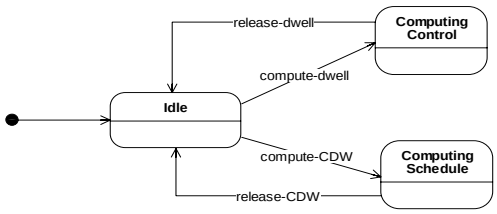

Fig. 7. The Computer

deteriorates with time. We ran numerous simulations of this kind. In most cases the result was that the system was unstable. Only for special settings of the K_p parameters could we gurantee the stability of the whole system.

Acknowledgments

This research was partially supported by a grant from the Defense Advanced Research Projects Agency.

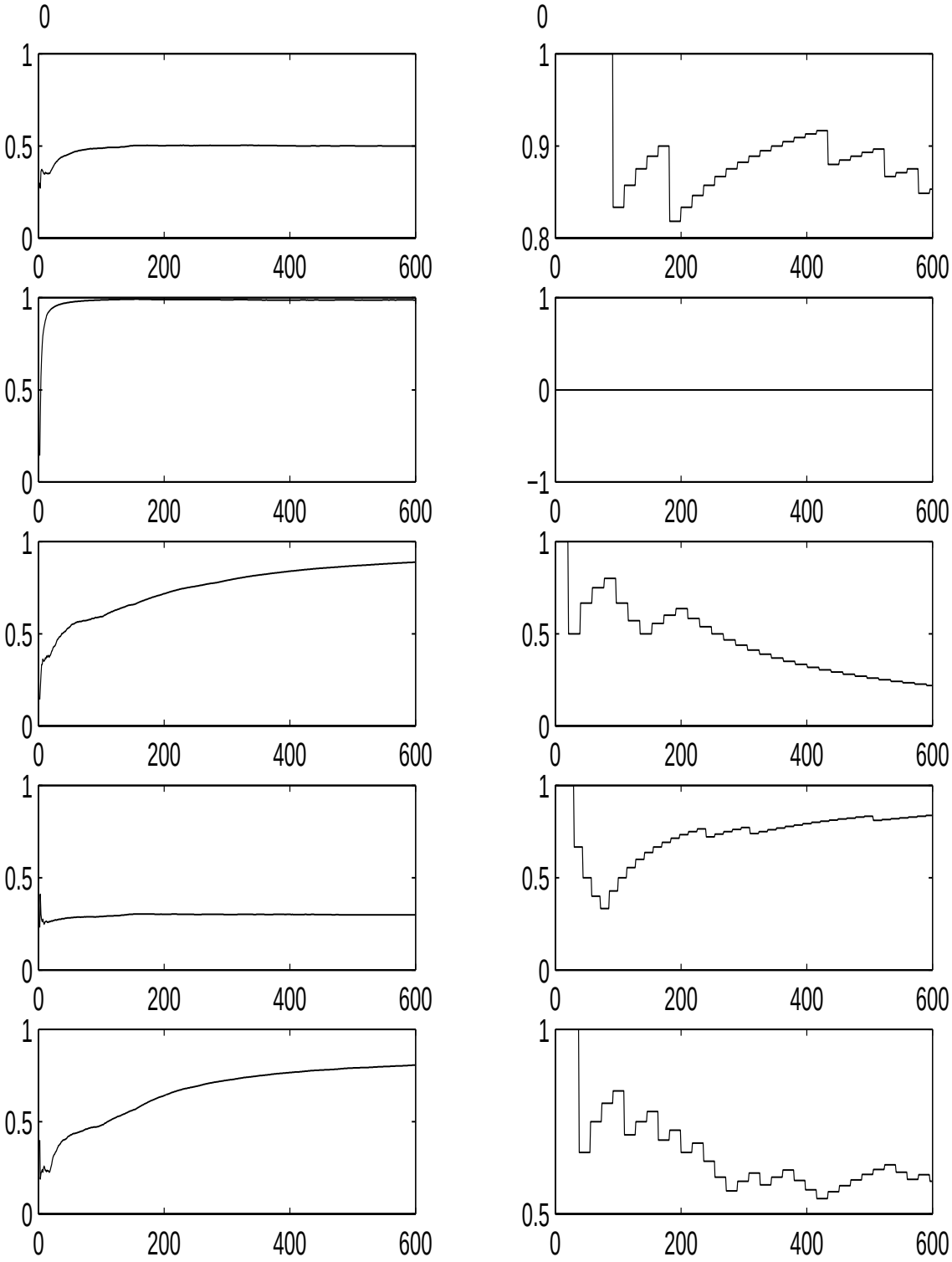

Fig. 8. Simulation Results for $K_p = 0.1$ for five radars. The left column shows Miss Rate (dwells requested by the controller, but missed by the scheduler. The right column represents probability of detection of radar illuminations.