

# Repair-Control of Enterprise Systems Using RFID Sensory Data

Wolf Kohn <sup>1</sup>

wk@hynomics.com

Hynomics Corporation, 3655 131st Av. SE. Suite 602, Bellevue, WA 98006

Dept. of Industrial Engineering, University of Washington, Seattle, WA.

Vladimir Brayman

vbrayman@hynomics.com

Hynomics Corporation, 3655 131st Av. SE. Suite 602, Bellevue, WA 98006

Dept. of Industrial Engineering, University of Washington, Seattle, WA.

Jana Littleton

janal@hynomics.com

Hynomics Corporation, 3655 131st Av. SE. Suite 602, Bellevue, WA 98006

Dept. of Industrial Engineering, University of Washington, Seattle, WA.

<sup>1</sup>The corresponding author

## Abstract

This paper provides a framework for implementing real-time enterprise planning, scheduling and control processes based on information provided by RFID sensing systems. The proposed framework is based on optimal control algorithms, and interfaces with existing ERP infrastructure. The objective is to respond autonomously to changes in the enterprise using a feedback configuration that minimizes disruptions. RFID sensing systems have the potential to provide the real time data needed to implement enterprise feedback functionality. The central concept presented in the paper is a real time repair schema implemented in a distributed architecture, that utilizes dynamical models described by differential equations with piece-wise continuous solutions.

## 1 Introduction

The real-time data provided by RFID technology by tagging products and tracking their movements at every point along the supply-chain has the potential to impact the efficiency and speed of enterprise processes such as inventory control, automated sales, price determination, and marketing. RFID sensing systems, when implemented, will provide accurate time-stamped data about flows and state of the enterprise, see Sweeny (2003) and Samuel (2003). In principle, the information generated from RFID sensing should significantly decrease the time dependent uncertainty of the state of the enterprise. This could allow for less conservative control strategies.

In order to realize its potential, RFID-based systems must overcome some important technological obstacles. One of these obstacles is that the amount of information generated by an RFID system grows exponentially with the number of different tagged items. For practical applications in enterprise process automation, this will require new approaches for real-time, distributed data handling and processing, Sweeny (2003). Another obstacle is that the detection processes for accurate discrimination, specifically the ability to accurately recognize tag information of multiple items, have not been completely solved.

In order to implement real-time autonomous enterprise control systems, sensing technologies must be significantly augmented with a decision-making system that *processes* the information extracted from the sensors and *encodes* it in some on-line database, *fuses* the information, see Kohn and Remmel (1996), James et al (1995), and *generates actions* that *automate* basic components of the enterprise operations.

Over the last two decades, enterprises implemented ERP systems for managing and automating business processes, for example Curran et al (1998). These systems constitute a significant investment in software, hardware, and training. Current ERP systems are not well suited to implement feedback systems with the functionalities mentioned above.

In this paper we describe real-time planning scheduling and control feedback systems that improve the performance of the enterprise by generating plans, schedules and actions that depend on real-time information from RFID and other sensing technologies (e.g. bar code systems). A *distributed feedback system* that realizes this approach must involve a global mapping from sensory data and strategy into actions. We refer to systems of this type as Enterprise Feedback Systems (EFS), see Figure 1.

When deployed EFS will dramatically increase the efficiency and performance in application areas such as capacity planning, scheduling, inventory control, sale and pricing automation, and other logistics areas. The expected substantial behavior improvement of the enterprise is a consequence of the significant reduction in operational enterprise uncertainty and of more accurate market forecasting. These systems are discussed in Section 2.

The central concept discussed in this paper is a *repair* implementation of EFS, Kohn and Brayman (2003), which is based on an optimal control architecture, Kohn et al (2003). A repair implementation of EFS corrects actions generated by a current ERP system to tune the enterprise process in response to real-time sensor data. Repair is needed to minimize *disruption* in the behavior of the enterprise due to unexpected events or a change in trend data such as demand. We will discuss this concept in more detail in Section 3.

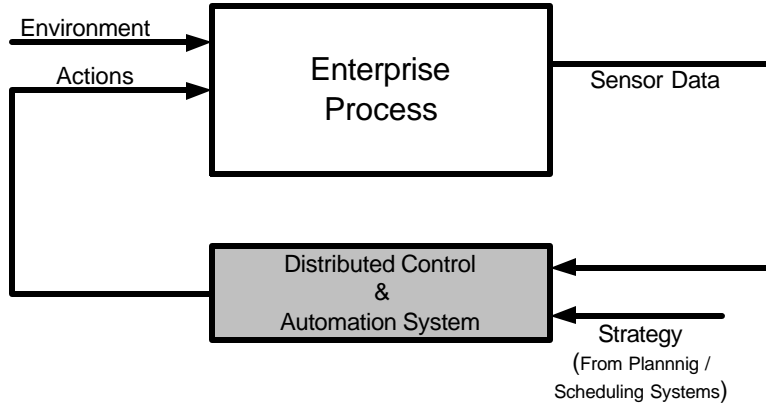


Figure 1: Control and Automation System

The central element of Control Repair Architecture is the Control Generator. We will describe the functionality of this element in Section 4. Finally, we present a distributed implementation of enterprise repair systems in Section 5. The enterprise models used for implementing the proposed architecture are continuous time models described by piecewise continuous differential equations.

We conclude this introductory section with a very important observation about the nature of models used to implement our enterprise control architecture. In order to implement a repair system continuous-time models represented by ordinary differential equations are better suited than discrete or heterogenous models. This is not a limitation because of an existing technology called *continualization* described in Kohn et al (1996 a), Kohn et al (1997), Kohn et al (1996 b), and Kohn et al (1989), which allows us to exactly encode discrete and rule-based model components as continuous and differential constraints.

## 2 Enterprise Feedback System

RFID and other real-time sensory data could provide a substantial enhancement in the performance of the enterprise brought by the self-correcting and stable tuning characteristics of a suitably designed feedback control system. Unfortunately, current ERP systems are

inadequate for generating real-time actions as a function of the very high volume of sensory data, Sweeny (2003).

In this section we describe the characteristics of the components of the proposed architecture. Figure 2 illustrates an enterprise process controlled by an ERP system and enhanced by a sensor system driving a real-time repair process. The repair components enable the following four improvements:

1. Enhancement of the performance of the enterprise.
2. Adequate response to the high-speed sensory data such as RFID.
3. Tuning the parameters of the enterprise model as a function of real-time sensor data.
4. Correction of planning, scheduling, and control functionality of the ERP system in response of the enterprise to unexpected events and other time-dependent changes.

In Figure 2, the architectural components labeled as ERP represent an existing automated enterprise management system (see Bowersox et al (1996)). The ERP system is a semi-autonomous system that generates, with different levels of abstraction, plans, schedules, and enterprise control actions for managing the enterprise. This type of system is described in Curran et al (1998) and O’Leary (2000).

In Figure 2, the block labeled Sensor System represents not just sensing elements but also data processing components. The planner generates a plan to implement an enterprise strategy. The scheduler assigns resources to execute the plan in the form of a schedule of activities, and the control and automation system generates actions to realize the schedule. Usually, because of inaccuracy in the modeling of the enterprise and noise in the sensory data, the plan can only be executed approximately at best. In many instances, continuous user intervention is needed to maintain adequate behavior of the enterprise, Lee et al (1997). The proposed feedback control system addresses these issues by generating stable control laws.

The networked components indicated with darkened lines are the elements of the proposed repair system. It includes three components, represented by the ovals in the diagram. In addition, the forecast element that is usually run independently, is incorporated into the loop.

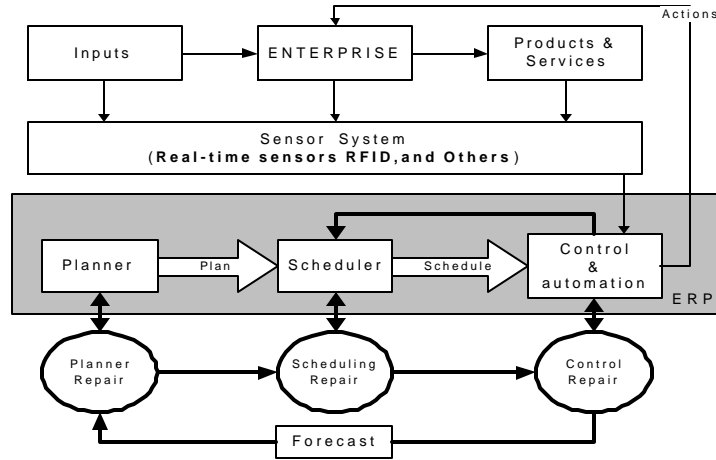


Figure 2: Enterprise Feedback Management Planning, Scheduling, and Control

In this diagram the enterprise block represents a process or processes that transforms the input (raw material, capital, labor, etc.) into product and services in a dynamic fashion. The model of this transformation will be referred to as Reference Model in Section 3.

Although Planning, Scheduling, and Control are very different functionalities, the underlying architectures of the Planner Repair, Scheduling Repair, and Control Repair components are very similar. Therefore the remainder of the paper focuses on the Control Repair component only.

We mention in concluding this section, that the structure of the planner and scheduling repair elements is very similar. We also would like to state that the diagram depicted in Figure 2 is a functional representation of the architecture. An operational representation, which is a distributed implementation, will be discussed in Section 5.

### 3 Control Repair Architecture

The control repair element has a distinctive architecture based on principles of Control Theory of dynamical systems with high level of uncertainty. The proposed architecture is a modified version of the so-called “model following” control law, see Astrom et al (1995). The modification is that the control law follows a model that is dynamically modified as a function of sensory data. The model is continuously tuned with respect to its parameters (parameter adaptation) and its structure (learning). This is essential for implementing automated schemas for enterprise systems which are modeled largely by empirical principles with highly uncertain parameters.

A diagram of the proposed control repair architecture is shown in Figure 3. This architecture consists of six elements: Control Generator, State Estimator, Reference Model, Adapter, Learning Engine, and Command Translator. The Command Translator is an element that translates dynamic actions into commands for the reference model, which is a real-time continuous simulation of the repairable dynamics of the enterprise.

Each of the other five elements is formulated as a dynamic optimization whose purpose is to compute in real-time the input-output map associated with the element. For example, the state estimator element computes and dynamically updates a map that generates an estimate of the repair state of the enterprise as a function of current sensor data, internal state estimate, actions, and model and parameter updates. Similarly, the control generator computes the control law map that generates actions as a function of actual output, simulated output, and parameter and model updates.

We will describe this schema only for the control generator element. We note that the behavior of *each* of the other elements is determined by the same generic optimization formulation. However the overall repair control behavior must be a solution of an optimization problem of minimizing the enterprise disruption with respect to the reference model, subject to *fence* constraints and operational constraints. Fence constraints are essential for implementing the repair schema and will be explained in Section 4.

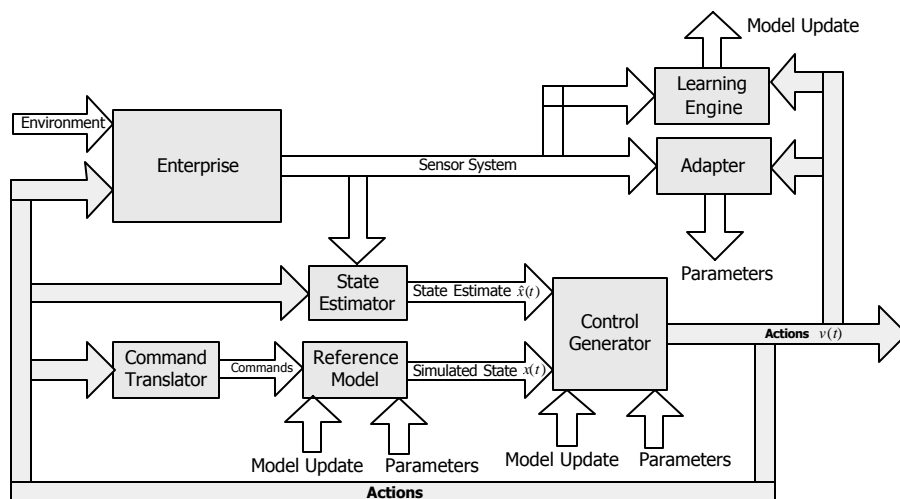


Figure 3: Repair Control Architecture

A map *separation principle* allows us to formulate the functional architecture shown in Figure 3. The separation principle provides a formal procedure for *decomposition* of the functionality of the overall minimum disruption optimization map into separate map optimizations for each element. The decomposition is important due to several reasons:

1. Scalability allows to decompose the optimization problem into several sub-problems whose additive complexity is lower than that of the original optimization.
2. Design. The decomposition facilitates the design of the distributed Control Architecture Node described in Section 5.
3. Stability. The separation principle guarantees stabilizability if the original functional architecture is stabilizable.
4. Functionality of each element can be approximated arbitrarily close in a distributed architecture.

This separation principle is based on the theory of harmonic morphisms and is described in a paper in preparation: Kohn and Brayman and also in Baird et al (2003).

Each of the elements in the architecture depicted in Figure 3 requires a separate treatment and we intend to do this in future papers. We conclude this section with a short summary of the functionality of each of the elements in the repair control architecture.

- Reference model: simulates the dynamic representation of the enterprise process under control (see Kohn et al (1994), Lee et al (1994), Nerode et al (1992), Nerode et al (1993)).
- Control Generator: computes the optimal control law as a causal map from the performance space ( the product of the sensor space , the parameter space the model frame space and the state space) to the space of actions.
- Adapter: computes optimal parameter estimates as a map from the space of observables and commands onto the space of parameters.
- Learning Engine: computes incremental optimal updates to the reference model based on causal observations and commands.
- State Estimator: computes optimal causal incremental estimates (forecast) of the state of the enterprise from sensor data. As parameters of the reference model are updated, the state estimator map is tuned.
- Command Translator: translates commands into actions.

## 4 Control Generator

The Control Generator implements a robust repair control map that produces actions in response to a wide range of events and variations encountered in the dynamics of enterprise processes. It generates time dependent action vector  $v(t)$  at each time  $t$  as a function of the estimated state trajectory  $\hat{x}(t)$ , and the simulated trajectory  $x(t)$ , see Figure 3. In this paper

we do not discuss the properties (continuity, convergence, existence, etc.) of the differential models, for details see Kohn and Remmel (1997).

The Control Generator operates in two modes: *reference* and *repair*. Both are implemented in a schema referred to as a sliding window mechanism consisting of a *window* of width  $T$  with time increment  $\Delta T$ , see Figure 7. The sliding window mechanism allows the *current* value of the action variables  $v(t)$  to depend on the simulated state of the enterprise generated by the reference model element and will be described later on in this section.

The reference mode is formulated as an optimization problem with criterion that defines the desired behavior of the enterprise process under control. This mode is needed for “cold starting” the process or when the state trajectory  $x$  and control trajectory  $v$  of the enterprise process from the previous horizon  $[t_1 - \Delta T, t_1 - \Delta T + T]$  are not available or are highly corrupted.

As we described before, the repair mode is applicable when one of the objectives is to minimize disruption with respect to previously computed enterprise state trajectory and control. These trajectories are generated in real time by the reference model element. We briefly describe the reference mode in the next subsection.

## 4.1 Reference Mode

The reference mode is characterized by a control optimization problem, Athans et al (1966). The moving time interval of this problem is the interval  $[t_1, t_1 + T]$ . The symbol  $t_1$  denotes current time generated by the real time “clock”. The symbol  $T$  denotes a constant: the width of the window.

The control optimization problem, whose solution is the reference mode state and action trajectories is given by:

$$\min_{v(t)} \int_{t_1}^{t_1+T} \tilde{\phi}(x(t), v(t), d(t)) dt \quad (1)$$

subject to

$$x_i(t) - x_i(t_1) - \int_{t_1}^t \tilde{f}_i(x(\tau), v(\tau)) d\tau = 0, \quad i = 1, \dots, n-1 \quad (2)$$

$$g_j(x(t), v(t)) \geq 0, \quad j = 1, \dots, m \quad (3)$$

$$v \in \mathcal{V} \quad (4)$$

$$t \in [t_1, t_1 + T], \quad (5)$$

where  $\mathcal{V}$  is a compact subset of  $R^r$  and  $d(t)$  is a driving function (e.g. deterministic or forecasted demand). Here  $\tilde{\phi}$ ,  $\{\tilde{f}_i, i = 1, \dots, n-1\}$ , and  $\{g_j, j = 1, \dots, m\}$  are sufficiently smooth.

We introduce a scalar variable  $x_n(t) \geq 0$  such that

$$x_n(t) = \int_{t_1}^t \tilde{\phi}(x(\tau), v(\tau), d(\tau)) d\tau, \quad (6)$$

$$x_n(t_1) = 0. \quad (7)$$

Then problem (1)-(4) becomes:

$$\min_{v(t)} x_n(t_1 + T) \quad (8)$$

subject to (for  $t \in [t_1, t_1 + T]$ )

$$x_i(t) - x_i(t_1) - \int_{t_1}^t f_i(x(\tau), v(\tau), d(\tau)) d\tau = 0, \quad i = 1, \dots, n \quad (9)$$

$$g_j(x(t), v(t)) \geq 0, \quad j = 1, \dots, m \quad (10)$$

$$v \in \mathcal{V}, \quad (11)$$

where

$$f_i = \begin{cases} \tilde{f}_i(x(t), v(t)) & \text{for } i = 1, \dots, n-1 \\ \tilde{\phi}(x(t), v(t), d(t)) & \text{for } i = n \end{cases} .$$

The solution of this problem,  $v^*(t)$ , is called the optimal control and the corresponding trajectory,  $x^*(t)$ , i.e. a trajectory that satisfies (9)-(11) and (8), is called the optimal trajectory. We solve problem (8)-(11) via a direct transcription formulation presented in Kohn and Brayman (2003). Notice that the solution to problem (8)-(11) gives the expected state trajectory  $x(t)$  and control trajectory  $v(t)$ ,  $t \in [t_1, T)$ , which is the output of the reference model. In this problem, the criterion is defined by the user.

## 4.2 Repair Mode

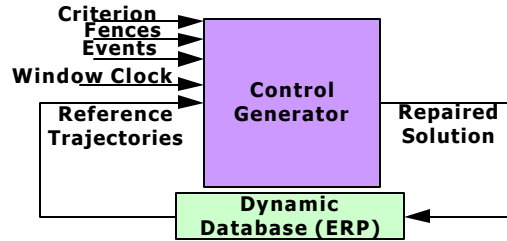


Figure 4: Repair Mode

Figure 4 shows a configuration for running a repair strategy which also uses a sliding window schema. The Control Generator receives as inputs a Criterion, Fences, Events, Window clock, and the Reference state and action Trajectories on the current window interval. The criterion used by the Control Generator may change to respond to unexpected situations. Therefore, it allows for the criterion to be changed during a window interval.

Fences are sub-intervals of the width of the window interval on which the state and/or action trajectories are not allowed to change with respect to reference trajectories. This is the mechanism used in our proposed implementation to indicate to the control generator what parts of the behavior of the enterprise during the current window interval are not allowed

to be disrupted. For example, the job assignments may be fenced in order to prevent widespread reorganization in the current enterprise execution policy.

Events are input variations that are accumulated at a single time point  $t_1$ , see Figure 5.

The reference trajectories in Figure 4 are described later on in this section.

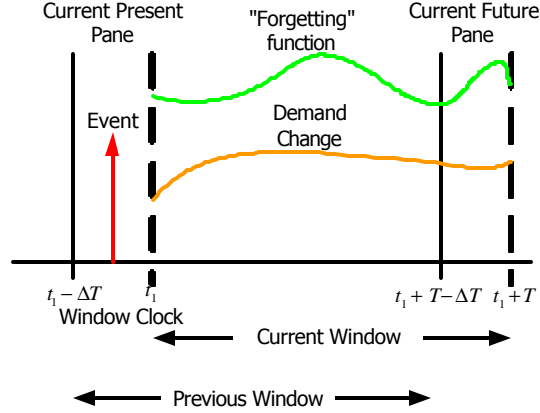


Figure 5: Sliding Window Framework

Let  $T$  be the width of the window, let  $\Delta T$ , the window update interval, be the duration of a window *pane*. A pane is the interval  $[t_1 - \Delta T, t_1]$  where  $t_1$  is the current time. During the pane the Control Generator element acquires the events accumulated in the interval  $[t_1 - \Delta T, t_1]$ , see Figure 5. These events are represented as an impulse at  $t_1$ .

The “Forgetting” function, Figure 5, is a continuous function over the window interval that allows the control generator to emphasize or dampen the time-dependent criterion to respond to current needs of the enterprise.

At time  $t_1 - \Delta T$  the reference control  $v^{*(t_1 - \Delta T)}(t)$  and corresponding reference trajectory  $x^{*(t_1 - \Delta T)}(t)$  for  $t \in [t_1 - \Delta T, t_1 - \Delta T + T]$  are computed (solid lines in Figure 7). These control and state trajectories are given at time  $t_1$ .

At time  $t_1 - \Delta T$  the window slides  $\Delta T$  units. The repair mode then has to compute the incremental changes in the control  $\delta v^{t_1}(t)$  and state  $\delta x^{t_1}(t)$  trajectories, for  $t$  in the time interval  $[t_1, t_1 + T]$ . These incremental changes are caused by five reasons:

1. As the window slides by  $\Delta T$  units, the time horizon of the optimization slides by  $\Delta T$

units so the trajectories are perturbed by a change in the horizon from  $t_1 - \Delta T + T$  to  $t_1 + T$ ,

2. The accumulation of events in the interval  $[t_1 - \Delta T, t_1]$ ,  $E\delta(t - t_1)$  to the system,
3. The time dependent parameters such as demand change in the interval  $[t_1, t_1 + T]$  see Figure 5,
4. The optimality criterion is allowed to change to accommodate unexpected situations (e.g. as a response to change in competitors' advertising strategy), see Figure 6. The Forgetting function is needed here to ensure the smooth and stable transition between criteria.
5. The previously set fences may change.

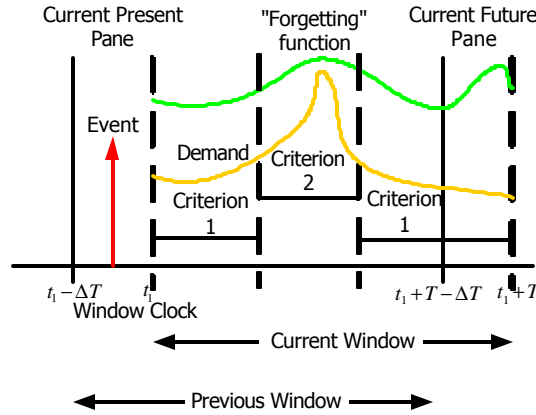


Figure 6: Criterion Switch

The control generator can handle two types of events: catastrophic and repairable. Catastrophic events cause the model to be rebuilt completely and the process to start anew using the Reference Model as described in the previous subsection. In this subsection we consider repairable events, that is events that cause changes in the model that can be represented by a modified perturbation model described in Kohn and Brayman (2003). Here,

repairable events are restricted in amplitude and interval of impact. In particular, we consider only those events that can be modeled as a term  $A^e \delta(t - t_e)$  added to the dynamics, where the event time  $t_e \in [t_1 - \Delta T, t_1]$ , the positive amplitude  $A^e \in [A_{\min}, A_{\max}]$ , and  $\delta(t)$  is the delta-distribution (impulse).

In the repair mode, the reference action and state trajectories are given.

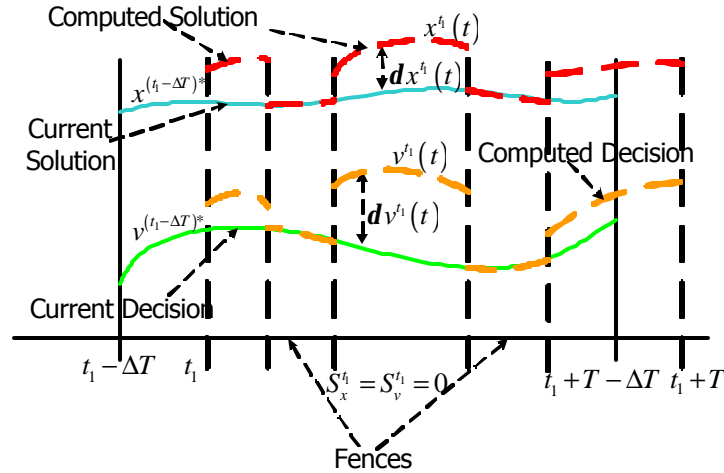


Figure 7: Sliding Window

Let  $v^{(t_1 - \Delta T)*}(t)$  and  $x^{(t_1 - \Delta T)*}(t)$  be the reference control and reference state trajectory respectively on the interval  $[t_1 - \Delta T, t_1 + T - \Delta T]$  computed at  $t_1 - \Delta T$ . Define  $\delta x_i^{t_1}(t)$  and  $\delta v_k^{t_1}(t)$  to be a state trajectory and control repair respectively.

Fences are time sub-intervals, where no deviation from the reference trajectory is allowed. Let  $S_{x_i}^{t_1}(t)$  and  $S_{v_k}^{t_1}(t)$  be the “fencing indicator” functions defined as follows

$$S_{x_i}^{t_1}(t) = \begin{cases} 0 & \text{if } t \text{ is in the “fenced” interval for } x_i \\ 1 & \text{otherwise} \end{cases}$$

$$S_{v_k}^{t_1}(t) = \begin{cases} 0 & \text{if } t \text{ is in the “fenced” interval for } v_k \\ 1 & \text{otherwise} \end{cases} .$$

We assume that there is a finite number of fenced intervals in  $[t_1, t_1 + T]$  specified by the

user, see Figure 7.

Then (for  $i = 1, \dots, n$ ), where  $n$  is the dimension of the model, the incremental state trajectory  $\delta x_i^{t_1}(t)$  starting at  $t_1$  is given by

$$\delta x_i^{t_1}(t) = S_{x_i}^{t_1}(t) \left\{ \begin{array}{l} x_i^{t_1}(t_1) - x_i^{(t_1 - \Delta T)^*}(t_1) - A_i^{t_1} \\ + \int_{t_1}^t \left[ \begin{array}{l} \sum_{j=1}^n \frac{\partial f_i}{\partial x_j} ((t_1 - \Delta T)^*) S_{x_j}^{t_1}(\tau) \delta x_j^{t_1}(\tau) \\ + \sum_{k=1}^r \frac{\partial f_i}{\partial v_k} ((t_1 - \Delta T)^*) S_{v_k}^{t_1}(\tau) \delta v_k^{t_1}(\tau) \\ \sum_{l=1}^n \frac{\partial f_i}{\partial d_l} ((t_1 - \Delta T)^*) \delta d_l^{t_1}(\tau) \end{array} \right] d\tau \end{array} \right\}, \quad (12)$$

where the notation  $(t_1 - \Delta T)^*$  means that the partial derivatives are evaluated along the reference trajectory, e.g.  $\frac{\partial f_i}{\partial x_j} ((t_1 - \Delta T)^*) = \frac{\partial f_i}{\partial x_j} (x^{(t_1 - \Delta T)^*}(t), v^{(t_1 - \Delta T)^*}(t), d^{(t_1 - \Delta T)^*}(t))$ .

The set  $\{ [t_s^{i,a}, t_s^{i,b}] , s = 1, \dots, M_i \}$  are the fenced time sub-intervals for state  $i = 1, \dots, n$ .

We approximate all the events  $A_i^e \delta(t - t_e)$  that happen during the  $[t_1 - \Delta T, t_1]$  time interval with a single event  $A_i^{t_1} \delta(t - t_1)$  at  $t_1$ , where  $A_i^{t_1}$  is the sum of all  $A_i^e$ . Then

$$\int_{t_1^-}^{t_1^+} e_i(\tau) d\tau = \int_{t_1^-}^{t_1^+} A_i^{t_1} \delta(\tau - t_1) d\tau = A_i^{t_1}.$$

The identity  $\delta x_i^{t_1}(t_1) = x_i^{t_1}(t_1) - x_i^{(t_1 - \Delta T)^*}(t_1) - A_i^{t_1}$  gives the initial conditions for  $\delta x_i^{t_1}(t)$ .

### 4.3 Analytic extension of the reference trajectory

Notice  $x_i^{(t_1 - \Delta T)^*}(t)$  and  $v_i^{(t_1 - \Delta T)^*}(t)$  are defined on  $[t_1 - \Delta T, t_1 + T - \Delta T]$ . In order to find the dynamics of  $\delta x_i^{t_1}(t)$  on this interval, we need to extend  $x_i^{(t_1 - \Delta T)^*}(t)$  and  $v_i^{(t_1 - \Delta T)^*}(t)$  to the time interval  $[t_1, t_1 + T]$ . We use the analytic extension, that is we assume that the continuation of a curve  $x_i^{(t_1 - \Delta T)^*}(t)$  on the interval  $[t_1 + T - \Delta T, t_1 + T]$  is due to a constant control, fixed at  $t = t_1 + T - \Delta T$ . We don't allow any "fencing" in  $[t_1 + T - \Delta T, t_1 + T]$ .

Then from (12),

$$\delta x_i^{t_1}(t) = \delta x_i^{t_1}(t_1 + T - \Delta T) + \int_{t_1 + T - \Delta T}^t \left[ \begin{array}{l} \sum_{j=1}^n \frac{\partial f_i^{(t_1 - \Delta T)^*}}{\partial x_j} (t_1 + T - \Delta T) \delta x_j^{t_1}(\tau) \\ \sum_{k=1}^n \frac{\partial f_i^{(t_1 - \Delta T)^*}}{\partial v_k} (t_1 + T - \Delta T) \delta \bar{v}_k^{t_1}(\tau) \\ + \sum_{l=1}^n \frac{\partial f_i^{(t_1 - \Delta T)^*}}{\partial d_l} (t_1 + T - \Delta T) \delta d_l(\tau) \end{array} \right] d\tau$$

$$t \in [t_1 + T - \Delta T, t_1 + T], \quad i = 1, \dots, n,$$

where  $\delta \bar{v}^{t_1}(t) = v(t_1 + T) - v^*(t_1 + T - \Delta T)$ .

Then the dynamics on the interval  $[t_1, t_1 + T]$  becomes

$$\delta x_i^{t_1}(t) = \left\{ \begin{array}{l} S_{x_i}^{t_1}(t) \left( \begin{array}{l} \delta x_i^{t_1}(t_1) \\ + \int_{t_1}^t \left[ \begin{array}{l} \sum_{j=1}^n \frac{\partial f_i}{\partial x_j} ((t_1 - \Delta T)^*) S_{x_j}^{t_1}(\tau) \delta x_j^{t_1}(\tau) \\ + \sum_{k=1}^n \frac{\partial f_i}{\partial v_k} ((t_1 - \Delta T)^*) S_{v_k}^{t_1}(\tau) \delta v_j^{t_1}(\tau) \\ + \sum_{l=1}^n \frac{\partial f_i}{\partial d_l} ((t_1 - \Delta T)^*) \delta d_l^{t_1}(\tau) \end{array} \right] d\tau \end{array} \right), \\ t \in [t_1, t_1 + T - \Delta T] \\ \delta x_i^{t_1}(t_1 + T - \Delta T) + \int_{t_1 + T - \Delta T}^t \left[ \begin{array}{l} \sum_{j=1}^n \frac{\partial f_i^{(t_1 - \Delta T)^*}}{\partial x_j} (t_1 + T - \Delta T) \delta x_j^{t_1}(\tau) \\ + \sum_{k=1}^n \frac{\partial f_i^{(t_1 - \Delta T)^*}}{\partial v_k} (t_1 + T - \Delta T) \delta \bar{v}_k^{t_1}(\tau) \\ + \sum_{l=1}^n \frac{\partial f_i^{(t_1 - \Delta T)^*}}{\partial d_l} (t_1 + T - \Delta T) \delta d_l(\tau) \end{array} \right] d\tau, \\ t \in [t_1 + T - \Delta T, t_1 + T] \end{array} \right. \quad (13)$$

#### 4.4 Repair Criterion

Criterion (8) more generally can be written as  $\Phi(x(t_1 + T))$ . The expansion up to the second order gives the repair criterion

$$\frac{1}{2} (\delta x(t_1 + T))^T \Phi_{xx}(x(t_1 + T)) \delta x(t_1 + T). \quad (14)$$

The goal of repair is to minimize criterion (14) while minimizing the change with respect to a nominal trajectory. Then the combined criterion is

$$\frac{1}{2} (\delta x^{t_1}(t_1 + T))^T \Phi_{xx}(x(t_1 + T)) \delta x^{t_1}(t_1 + T) + \frac{1}{2} (\delta x^{t_1}(t))^T Q \delta x^{t_1}(t) + \frac{1}{2} (\delta v^{t_1}(t))^T R \delta v^{t_1}(t), \quad (15)$$

where  $Q$  and  $R$  are constant positive definite matrices specified by a user that define a balance between satisfaction of the enterprise criterion (14) and minimizing the change.

## 4.5 Repair Constraints

In this short subsection we compute the constraints for the repair problem based on the constraints of the reference mode. From (3),

$$g_j(x^{(t_1 - \Delta T)^*}(t) + \delta x^{t_1}(t), v^{(t_1 - \Delta T)^*}(t) + \delta v^{t_1}(t)) \geq 0, \quad j = 1, \dots, m. \quad (16)$$

We expand (16) up to the first order and obtain,

$$\begin{aligned} g_j(x^{(t_1 - \Delta T)^*}(t) + \delta x^{t_1}(t), v^{(t_1 - \Delta T)^*}(t) + \delta v^{t_1}(t)) &\approx g_j(x^{(t_1 - \Delta T)^*}(t), v^{(t_1 - \Delta T)^*}(t)) \\ &+ \frac{\partial g_j}{\partial x}(x^{(t_1 - \Delta T)^*}(t), v^{(t_1 - \Delta T)^*}(t)) \delta x^{t_1}(t) + \frac{\partial g_j}{\partial v}(x^{(t_1 - \Delta T)^*}(t), v^{(t_1 - \Delta T)^*}(t)) \delta v^{t_1}(t) \\ \text{a.e., } j = 1, \dots, m. & \end{aligned} \quad (17)$$

In order to assure that inequality (16) is satisfied, given that

$$g_j(x^{(t_1 - \Delta T)^*}(t), v^{(t_1 - \Delta T)^*}(t)) \geq 0, \quad j = 1, \dots, m,$$

the second and the third terms on the right-hand side in (17) must satisfy

$$\frac{\partial g_j}{\partial x} (x^{(t_1-\Delta T)^*}(t), v^{(t_1-\Delta T)^*}(t)) \delta x^{t_1}(t) + \frac{\partial g_j}{\partial v} (x^{(t_1-\Delta T)^*}(t), v^{(t_1-\Delta T)^*}(t)) \delta v^{t_1}(t) \geq 0 \quad (18)$$

*a.e.*,  $j = 1, \dots, m$ .

The inequalities (18) are determined by perturbation from the original constraints of the optimization formulation in repair mode.

We note that  $\delta x^{t_1}(t)$  is discontinuous. Therefore the expansion (17) is piecewise continuous.

The repair optimization formulation is given by: minimize criterion (15), subject to (13) and (18).

Notice also that  $\delta v^{t_1}(t)$  must be such that  $v^{(t_1)^*}(t) \in \mathcal{V}$ .

## 5 Distributed Implementation

In Section 4 we presented a *functional architecture* for implementation of our proposed repair schema. In this section we discuss a distributed *implementation architecture*. The distributed architecture consists of a network of operational *clusters* running asynchronously. Figure 8 shows an operational diagram of a single cluster. The components of the cluster retain the functionality of the architecture depicted in Figure 3. In particular, state estimator and adapter in the cluster implement in a distributed fashion the functions of the State Estimator, Reference Model, and Adapter elements similar to the one described in Section 3. However, the functionality of the Learning Engine element is realized by a combination of the adapter and a component labeled Tuning Controller whose function is to generate actions that drive the Enterprise Component under control to compensate for the observed differences between the Simulated State and the Estimated State, see Figure 8.

The repair dynamics is implemented by the Repair Controller. This controller executes the reference and repair modes discussed in Section 4 in a distributed form. The structure of

this procedure is similar to the one presented in the previous section, but in both the reference and the repair modes, synchronization of the network of clusters is achieved *implicitly* by modifying the criteria and by adding a *synchronization constraint*.

The criteria modification consists of adding a term that is a function of the synchronization action variables. This term becomes very large relative to the other terms if the Synchronization Controller detects that the cluster is out of synch with the Node Network. The constraint that is added to the optimization problem in each mode is designed to force the generation of node solutions that are feasible with respect to the repair solution. It can be shown that, under very general conditions, this approach leads to an algorithm that reacquires and maintains synchronization in the presence of unexpected events (see Kohn et al, Kohn et al (2000), Kohn et al (1996 b), and Kohn et al (1992)). The proposed mechanism is sometimes referred to as the penalty method by other authors (e.g. Polak (1997)). We refrain from using this term because in addition to the criterion modification, we add an additional constraint to the problem formulation.

Actions generated by the Synchronization, Repair, and Tuning Controllers are *combined* by the Merger Controller. The combined action is feasible and achieves the optimal combination of three actions: repair action, synchro action, and correction action. The Merger Controller realizes time-division multiplexing of the three types of actions over  $\Delta T$  such that the mixture is optimal, see Ge et al (1996). This approach allows us to generate quasi-optimal solutions for each of the controllers in the node.

## 6 Conclusions

This paper presents a feedback repair system for implementing autonomy in enterprise processes to achieve significant improvement in enterprise productivity. In order to realize this improvement, we need real-time sensory data provided by RFID sensory systems to characterize the status of the enterprise. However, this is not enough. More fundamentally, we

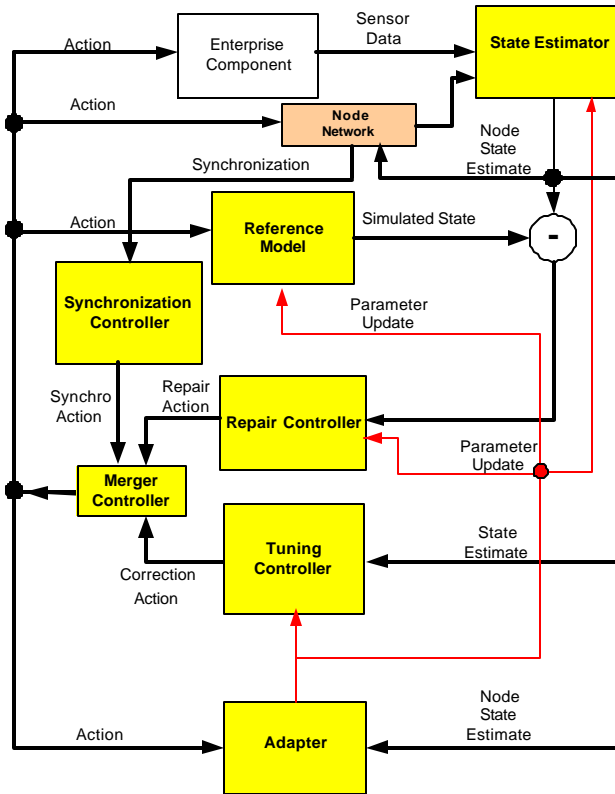


Figure 8: Distributed Control Architecture Node

need a framework that combines existing ERP infrastructure with a real-time repair system to generate control laws that drive the enterprise to achieve objectives that are not possible today.

The proposed repair system is formulated as a perturbation dynamics system. This means that both the action and the state trajectories of the enterprise are piecewise continuous. The operational criterion for the repair system is a combination of two components, the user-defined criterion for the enterprise operation and the disruption criterion. For practical reasons, minimizing disruption is an essential property, examples include: drastic change in schedules, machine assignments, sales strategies, routing, etc.

Repair systems have the ability to fence in time dependent state and action segments to generate solutions with acceptable levels of disruption with respect to a reference behavior. The proposed repair schema allows for dynamic modification of the criterion if needed to

respond appropriately to unexpected events. In order to accomplish appropriate response of the repair system, we need an architecture that supports repair and is compatible with the distributed nature of enterprise processes. We propose such an architecture in Section 5.

Although this paper only discusses in some detail the control repair component of the architecture the scheduling and planning repair components have been formulated similarly. We will report on particularities in future papers. Our research group at Hynomics Corporation is currently engaged in formulating applications to various enterprise problems based on the central ideas presented in this paper.

## References

- Astrom et al (1995)            Astrom, K.J. and Wittenmark, B. (1995) *Adaptive Control*, Addison-Wesley, Reading, MA.
- Athans et al (1966)            Athans, M. and Falb, P. (1966) *Optimal Control: An Introduction to the Theory and Its Applications*, McGraw-Hill, New York.
- Baird et al (2003)            Baird, P. and Wood, J.C. (2003) *Harmonic Morphisms Between Riemannian Manifolds*, Clarendon Press, Oxford.
- Bowersox et al (1996)        Bowersox, D.J. and Closs, D.J. (1996) *Logistical Management: The Integrated Supply Chain Process*, McGraw-Hill, New York.
- Curran et al (1998)         Curran, T. and Keller, G. (1998) *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*, Prentice Hall, Upper Saddle River, NJ.
- Ge et al (1996)                Ge, X., Kohn, W., Nerode, A. and Rummel, J.B. (1996) Hybrid Systems: chattering approximations to relaxed controls, *Hybrid*

- Systems III* (eds: Alur, R., Henzinger, T.A., Sontag, E.D.), Springer Lecture Notes in Computer Science, **1066**, pp. 76-100.
- James et al (1995) James, J., Nerode, A., Kohn, W., Cummings, B. and Chandra, J. (1995) Resources and Readiness: Issues in Evaluating Sensor Fusion Alternatives using a Cost Based Approach, *63rd Military Operations Research Society Symposium*.
- Kohn et al (1996 a) Kohn, W. and Rummel, J.B. (1996) Digital to Hybrid Program Transformations, *IEEE International Symposium on Intelligent Control - Proceedings 1996*, IEEE Press, Piscataway, NJ, pp. 342-347.
- Kohn et al (1997) Kohn, W. and Rummel, J.B. (1997) Hybrid dynamic programming, *Lecture Notes in Computer Science*, **1201**, pp. 391.
- Kohn et al (1996 b) Kohn, W., Nerode, A. and Rummel, J.B. (1996) Continualization: A Hybrid Systems Control Technique for Computing, *Proceedings of CESA '96 IMACS Multiconference*, pp. 517-521.
- Kohn et al (1989) Kohn, W., Jean Butler, J. and Graham, R. (1989) The Rational Tree Machine, *Proceedings of the SPIE-The International Society for Optical Engineering*, **1095**, pp. 264-274.
- Kohn and Rummel (1997) Kohn, W. and Rummel, J.B. (1997) Cost-Based Generation of Scalable, Reliable, Real-Time Software Components, *Hynomics Technical Report*, Hynomics Corporation, Bellevue, WA.
- Kohn et al (2003) Kohn, W., Brayman, V. and Nerode, A. (2003) Automated Sales and Supply Control of Enterprise Systems via Agent Cluster Networks, *IFAC Conference on Analysis and Design of Hybrid Systems*.

- Kohn et al (1994) Kohn, W., Remmel, J.B. and Nerode, A. (1994) Reactive Control of Distributed Interactive Simulations, *Simulation Journal*, **4**, pp. 382-398.
- Kohn and Brayman (2003) Kohn, W. and Brayman, V. (2003) Dynamic Problem, *Hynomics Technical Report*, Hynomics Corporation, Bellevue, WA.
- Kohn et al (2000) Kohn, W., Brayman, V. and Ritcey, J. (2000) Enterprise Dynamics via Non-Equilibrium Membrane Models, *Open Systems and Information Dynamics*, **7**, pp. 327-348.
- Kohn and Remmel (1996) Kohn, W. and Remmel, J.B. (1996) Scalable Data and Sensor Fusion via Multiple-Agent Hybrid Systems, Research supported by SDIO contract DAA H-04-93-C-0113, Hynomics Corporation.
- Kohn et al (1992) Kohn, W. and Nerode, A. (1992) Multiple Agent Autonomous Hybrid Control Systems, *Proceedings of the 31st Conference on Decision and Control*, pp. 2956-2966.
- Kohn and Brayman Kohn, W. and Brayman, V. (to appear) Optimal Feedback Control Law Generated via Harmonic Morphisms, *International Journal of Hybrid Systems*.
- Kohn et al Kohn, W., Brayman, V., Cholewinski, P. and Nerode, A. (to appear) Control in Hybrid Systems, *International Journal of Hybrid Systems*.
- Lee et al (1997) Lee, H., Padmanabhan, V. and Whang, S. (1997) Information Distortion in a Supply Chain: The Bullwhip Effect, *Management Science*, **43**, pp. 546-558.

- Lee et al (1994) Lee, T., Ghosh, S., Lu, J., Ge, X., Nerode, A. and Kohn, W. (1994) A Mathematical Framework for Asynchronous, Decentralized, Decision-Making Algorithm with Semi-Autonomous Entities: Synthesis, Simulation, and Evaluation, *Proceeding of 1994 Symposium on Intelligent Control*.
- Nerode et al (1992) Nerode, A. Kohn, W. (1992) An Autonomous Systems Control Theory: An Overview, *Proceedings of IEEE CACSD '92*, Napa, CA.
- Nerode et al (1993) Nerode, A. and Kohn, W. (1993) Multiple Agent Autonomous Control: A Hybrid Systems Architecture, *Logical Methods in Honor of Anil Nerode's Sixtieth Birthday* (eds: Crossely, N.C., Remmel, J.B. and Sweedler, M.E.), Birkhauser, Boston, 1993.
- O'Leary (2000) O'Leary, D.E. (2000) *Enterprise Resource Planning Systems: Systems, Life Cycle, Electronic Commerce, and Risk*, Cambridge University Press.
- Polak (1997) Polak, E. (1997) *Optimization: Algorithms and Consistent Approximations*, Springer, New York.
- Samuel (2003) Samuel, J. (2003) Bridging the Communications Gap Between Technologists and End Users in Radio Frequency Identification, Presentation at the 2003 RFID Workshop, University of Washington, Seattle, WA.
- Sweeny (2003) Sweeny, R. (2003) AutoID as a Disruptive Technology, Issues and Opportunities, Presentation at the 2003 RFID Workshop, University of Washington, Seattle, WA.