Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

PREVENTIVE WHAT-IF ANALYSIS IN SYMBIOTIC SIMULATION

Heiko Aydt
Stephen John Turner

School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

Wentong Cai
Malcolm Yoke Hean Low

School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

Peter Lendermann
Boon Ping Gan

D-SIMLAB Technologies Pte Ltd
9 Jurong Townhall Road #03-45 iHub
Singapore 609431

Rassul Ayani

School of Information and Communication Technology
Royal Institute of Technology (KTH)
Stockholm, Sweden

ABSTRACT

The what-if analysis process is essential in symbiotic simulation systems. It is responsible for creating a number of alternative what-if scenarios and evaluating their performance by means of simulation. Most applications use a reactive approach for triggering the what-if analysis process. In this paper we describe a preventive triggering approach which is based on the detection of a future critical condition in the forecast of a physical system. With decreasing probability of a critical condition, using preventive what-if analysis becomes undesirable. We introduce the notion of a *G*-value and explain how this metric can be used to decide whether or not to use preventive what-if analysis. In addition, we give an example for a possible application in semiconductor manufacturing.

1 INTRODUCTION

Symbiotic simulation is a paradigm which refers to a close relationship between a simulation system and a physical system and was originally defined at the Dagstuhl seminar on Grand Challenges for Modeling and Simulation in 2002 (Fujimoto et al. 2002). An essential concept of symbiotic simulation is the *what-if analysis process* (*WIA process*) which is concerned with the evaluation of a number of alternative what-if scenarios by means of simulation. As result of the WIA process, the best decision option is determined among the ones that were evaluated in respect to the evaluation criteria used.

An important issue which has to be considered when using symbiotic simulation is the triggering of the WIA process. The observation of a critical condition in the physical system is probably the most obvious reason for triggering a what-if analysis. In this case, the purpose of the WIA process is to find a solution to recover from the critical condition as quickly as possible. Another reason for triggering the WIA process could be the detection of a future critical condition in a forecast of the physical system. In this case, the purpose of the symbiotic simulation system is to prevent the critical condition or, at least, minimise its negative effects. We distinguish between these two triggering approaches and further refer to them as *reactive WIA* and *preventive WIA*, respectively.

The definition of a critical conditions is highly application specific. In addition, not all kinds of critical conditions can be forecasted. For example, sudden machine breakdowns cannot be forecasted. Using preventive WIA for this kind of conditions is therefore not possible and reactive WIA has to be used instead. However, some types of critical conditions can be forecasted. In this paper we will describe a possible application in semiconductor manufacturing where the queue length, in combination with a certain threshold, can be used for indicating a critical condition. In this application, the length of a queue can be forecasted and the application of preventive WIA is therefore possible.

It could be argued that preventive WIA performs always better than reactive WIA because it uses forecasting and can therefore react earlier. However, there is always a risk that the forecast is wrong. In this paper we describe two types of errors that have to be handled when using preventive

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

WIA. The notion of a *G*-value will be introduced which can be used to decide on the applicability of preventive WIA. Furthermore, we consider the application of symbiotic simulation in the context of semiconductor manufacturing and propose a simple error handling approach which combines reactive and preventive WIA.

The remainder of this paper is structured as follows: in Section 2 we discuss related work on symbiotic simulation with focus on what-if analysis. In Section 3 we explain reactive WIA and preventive WIA. In Section 4 we identify issues with preventive WIA, discuss two types of errors, and introduce the *G*-value metric. In Section 5 we describe a potential application of symbiotic simulation in semiconductor manufacturing and propose a simple error handling approach. In Section 6 we evaluate the performance advantage of preventive WIA in our application context. In Section 7 we present our conclusions.

2 RELATED WORK

Reactive WIA has been used to improve the performance of a semiconductor backend assembly and test facility (Low et al. 2005). A set of what-if scenarios is simulated in order to find optimal values for upper and lower queue sizes for a particular machine. These queue sizes are required to decide on outsourcing of lots to external vendors. The WIA process in this application is triggered if utilisation of machines exceeds a certain limit.

In our previous work we already used symbiotic simulation in the context of semiconductor manufacturing to automatically control a wet bench toolset (WBTS), using reactive WIA (Aydt et al. 2008). The what-if scenarios in that application represent alternative tool configurations and the WIA process is triggered once the queue length of pending lots has exceeded a certain threshold. We compared the performance of the symbiotic simulation control approach with the common practise control approach. Our results have shown that the performance of the WBTS can be significantly improved when using symbiotic simulation control.

Another example for reactive WIA can be found in (Low et al. 2007), where a symbiotic simulation system is used to monitor real-world operations and optimise a business workflow in the context of high-tech manufacturing and service networks. An aerospace spare components logistics show-case is described, where a WIA process is triggered if the fillrate performance of the physical system falls below an acceptable level. Each what-if scenario is using a different business workflow and the purpose of the symbiotic simulation system is to identify and implement the most suitable one, i.e., the one which produces the best fillrate performance among the evaluated scenarios.

Symbiotic simulation has also been used for UAV path planning (Kamrani and Ayani 2007). A set of alternative paths is created and evaluated by means of simulation. The purpose of these what-if simulations is to identify the best path for the UAV. Unlike other applications, the notion of a critical condition does not exist and the WIA process is triggered periodically. Therefore, the WIA process is triggered in a pro-active fashion in order to continuously improve the performance rather than reacting on a triggering condition.

Reactive and pro-active on-line planning has been discussed in the context of scheduling in manufacturing (Davis 1998). In this system, various alternative control policies are evaluated either at specific decision points (i.e., triggering in reactive fashion) or continuously (i.e., pro-actively) without being explicitly triggered.

According to (Aloulou and Portmann 2005) many forms of scheduling and rescheduling approaches have been proposed in the literature. These include reactive, predictive, pro-active, and pro-active/reactive scheduling approaches. In comparison with symbiotic simulation, reactive scheduling approaches can be compared with reactive WIA as both approaches aim to react upon a disruption in the physical system. However, the other approaches are difficult to compare with what-if analysis as they are specifically used to solve the scheduling problem.

3 WHAT-IF ANALYSIS

3.1 Reactive What-If Analysis

A reactive WIA is carried out if a critical condition is observed in the physical system. Once triggered, the WIA process has to make a decision regarding the physical system in order to recover from this condition as quickly as possible. In addition to identifying a solution it is also important to identify an appropriate action point, further denoted with t_a. With action point we refer to the point of time when a particular decision is implemented. There may be applications in which delayed implementation (i.e., $t_a > t_{now}$) results in a shorter recovery time. However, this case is not considered here as it probably represents a rare exception. Instead, we assume that it is an objective in reactive WIA to make a decision as quickly as possible.

Depending on how much time is required to make and implement a decision, we can consider $t_a \approx t_{now}$. This is based on the assumption that the WIA process can finish in a reasonably short period of time. This assumption is highly application-specific. If there is a significant delay until the earliest possible action point (i.e., if $min(t_a) >> t_{now}$), then it is necessary to consider this delay in the WIA process. In Section 5.2, we give an example from semiconductor manufacturing and explain why this assumption is valid in our application context. Reactive WIA is illustrated in Figure 1.

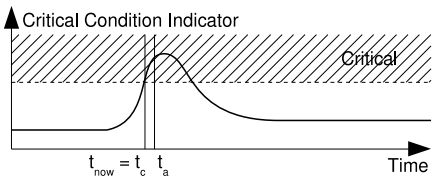

Figure 1: Conceptual illustration of reactive WIA.

3.2 Preventive What-If Analysis

A preventive WIA is carried out if a critical condition is detected in a forecast. For this purpose it is necessary to periodically perform forecasts of the physical system. Once a critical condition is detected, the WIA process is triggered. In addition, to finding an appropriate solution, it is also necessary to evaluate various action points. For example, consider the case in which a sudden performance drop is detected in the forecast of a manufacturing system in approximately eight hours from now. The control system may evaluate any possible action point until the physical system becomes critical at $t_c = t_{now} + 8h$. Appropriate preventive counter measures can take place at any time before t_c, i.e., $t_{now} \leq t_a < t_c$.

Depending on the application context, the exact timing might not be so important as long as appropriate counter measures take place before t_c. However, in many applications this is not the case but rather the exception. Counter measures which take place too early or too late may not adequately resolve the anticipated problem. In the worst case, bad timing might not help to resolve the problem at all or even make it worse. Since preventive WIA has to consider different action points in addition to finding an appropriate decision option, it is computationally more expensive than reactive WIA. Preventive WIA, using three action points, is illustrated in Figure 2.

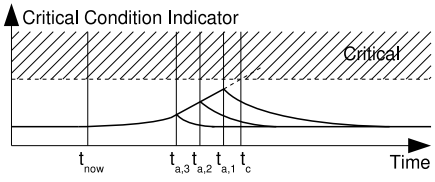

Figure 2: Conceptual illustration of preventive WIA.

4 IMPORTANT ISSUES IN PREVENTIVE WIA

An important issue when using preventive WIA is the ability to correctly forecast critical conditions. A forecast is never perfectly accurate and there will always be some uncertainty regarding whether or not the physical system will become critical. Therefore, there is a risk that a critical condition is not detected correctly. We consider the forecast as a hypothesis test to see whether there is enough evidence to support the hypothesis H_0 that there will be a critical condition in the physical system at some time t_c in future. The null hypothesis can either be true (i.e., the physical system actually becomes critical) or false (i.e., the physical system does not become critical). In addition, the forecast either detects a critical condition (i.e., fails to reject H_0) or fails to detect a critical condition (i.e., rejects H_0). This leads to the four cases illustrated in Table 1.

Table 1: Four different cases in preventive WIA.

	H_0 is true	H_0 is false
Detect	Case 1	Type II error
Fail to Detect	Type I error	Case 2

4.1 Different Types of Errors

In two of these four cases, the analysis of the forecast comes to the correct conclusion: either a critical condition was correctly detected (case 1) or nothing was detected because there will be indeed no critical condition (case 2). The other two cases represent errors which can lead to significantly reduced performance. These two types of errors are referred to as type I and type II error, respectively. In case of a type I error, the WIA process process will not be triggered and no counter measures will be taken in order to prevent the critical condition from occurring. In case of a type II error, the WIA process will be triggered based on wrong assumptions. As result, inappropriate counter measures are implemented in order to prevent the alleged critical condition from occurring.

We consider two types of errors which can occur during the process of detecting a critical condition. This implies that there is only one cause of a critical condition and the forecast can either detect it or not. Although this might be the case in some systems, there are certainly systems in which it is necessary to detect and handle differrent kinds of critical conditions. Consider the case where the *wrong* critical condition, say critical condition A, is detected while critical condition B will actually occur. It may seem as if this can be classified as a third type of error. However, this error can be described as a combination of a type I and a type II error. The system wrongly detected critical condition A (type II error) and failed to detect critical condition B (type I error).

These two types of errors have to be handled accordingly. Depending on the application context, one type of error may be more severe than the other. For example, in some applications taking unnecessary counter measures against a critical condition which does not occur might be considered as a lesser evil than failing to detect a critical condition which has devastating effects on the physical sys-

tem. This is the case for all safety related applications, for instance. In such an application, a critical condition may refer to physical damage or even life-threatening conditions. In other applications, a critical condition may refer to suboptimal performance which may ultimately lead to the interruption of a process flow. Therefore, the severeness of each error type has to be assessed individually and an appropriate error handling strategy has to be chosen for each type of application.

4.2 Applicability of Preventive WIA

Preventive counter measures should only be applied if there is enough evidence for the occurrence of the critical condition. A preventive WIA process is therefore triggered if the probability of a critical condition, further denoted with ε, is reasonably high. In order to make a decision regarding the application of preventive WIA, a critical propability level ε_c is used. If $\varepsilon \geq \varepsilon_c$ then there is enough evidence to justify the application of preventive WIA. If ε_c is too low or high then the triggering of the WIA process is either hasty or overly cautious, respectively. An appropriate choice of ε_c is therefore crucial.

We propose a method which can be used to determine the critical probability level based on a performance metric which indicates how much better or worse preventive WIA is in comparison with reactive WIA. This metric is further referred to as *G-value* and represents a measure of the relative performance gain of preventive WIA to reactive WIA over a period of time. It is defined as

$$G = \frac{preventive(t_s, t_e)}{reactive(t_s, t_e)} \tag{1}$$

where $preventive()$ and $reactive()$ are functions used to determine how the corresponding triggering approach performed over the period under consideration. The assessment period is defined by starting time t_s and end time t_e. The application of preventive WIA is desirable for $G > 1$.

Specific G-values are used to represent the performance gain or loss for each of the four different cases illustrated in Table 1. In the two ideal cases, the presence or absence of a critical condition was correctly detected (case 1 and 2, respectively). The specific G-values associated with these cases are further denoted with G_γ and G_δ, respectively. If the absence of a critical condition was correctly forecasted, the preventive WIA approach will not implement any counter measures. The behaviour is therefore exactly the same as compared to the reactive WIA approach. The specific G-value for this case is therefore always $G_\delta = 1$. The specific G-values associated with type I and type II error cases are further denoted with G_α and G_β, respectively.

Whether or not to apply preventive WIA depends on the probability ε of an alleged critical condition and possible consequences in terms of performance gain/loss, expressed by the corresponding specific G-values. If counter measures *are* implemented the outcome is either a case 1 situation (G_γ applies) or a type II error situation (G_β applies). The overall G-value for this scenario is therefore:

$$G_1 = \varepsilon G_\gamma + (1-\varepsilon) G_\beta \tag{2}$$

If *no* counter measures are implemented the outcome is either a case 2 situation (G_δ applies) or a type I error situation (G_α applies). The overall G-value for this scenario is therefore:

$$G_0 = \varepsilon G_\alpha + (1-\varepsilon) G_\delta \tag{3}$$

The objective is to decide whether to implement counter measures or not, using 2 and 3. If $G_1 > G_0$, then the implementation of counter measures is desirable. With decreasing ε, the application of preventive WIA becomes less attractive. At some point it becomes undesirable to use preventive WIA. This point is the critical probability level ε_c which is defined as the intersection of G_1 and G_0:

$$G_1 = G_0 \Rightarrow \varepsilon_c = \frac{1 - G_\beta}{G_\gamma - G_\alpha - G_\beta + 1} \tag{4}$$

For example, consider the following example: let $G_\gamma = 1.23$, $G_\alpha = 0.9$, and $G_\beta = 0.8$. From (4) it follows that $\varepsilon_c = 0.38$. Therefore, the application of preventive WIA is reasonable if the probability of the critical condition is $\geq 38\%$. This is to ensure that using preventive WIA is beneficial in the long run. The corresponding G-values for G_1 and G_0 are illustrated in Figure 3 as functions over ε.

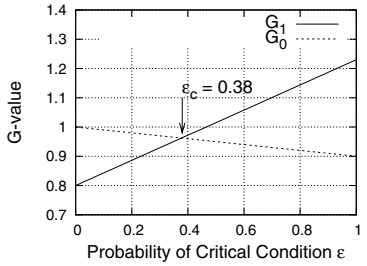

Figure 3: Example of intersection of G_1 and G_0 at the critical probability level $\varepsilon_c = 0.38$.

5 APPLICATION OF SYMBIOTIC SIMULATION

In this paper we consider the application of symbiotic simulation and preventive WIA for operational control of semiconductor manufacturing equipment. The semiconductor manufacturing process is very complex and asset intensive (Scholl and Domaschke 2000). Hence, it is important for semiconductor manufacturing companies to use their assets

more efficiently in order to reduce demand for additional equipment. In addition, it is important for them to continuously improve the performance of the manufacturing process in order to stay competitive. Current trends in the semiconductor industry show that automation solutions become increasingly important for improving efficiency and performance (Gan, Chan, and Turner 2006).

Various performance metrics are considered in semiconductor manufacturing. One of the most important ones (Pfund, Mason, and Fowler 2006) is the cycle time which is the period of time when a wafer enters a semiconductor manufacturing plant (fab) until it leaves the fab. There are always some delays until a wafer can be processed by a specific tool because the tool is busy, for instance. The cycle time is therefore always longer than the raw processing time. Another performance metric, which is often used, is the cycle time factor which is the ratio of actual cycle time and raw processing time. Here, we will always use the cycle time factor because it allows us to compare the performance for wafers with different process flows, i.e., wafers with different raw processing times.

5.1 Wet Bench Tool Set (WBTS)

A wet bench tool is used to clean a wafer after certain fabrication steps in order to remove particles which would otherwise compromise the quality of the wafer. Typically, there are several wet benches available in a fab. A wet bench contains a number of baths, each containing a particular chemical liquid. Wafer lots are processed in these baths strictly according to an associated recipe. A recipe defines the exact sequence of baths and the period of time a wafer lot needs to spend in a particular bath. Failing to comply with these constraints will reduce the quality of the wafer lot and possibly requiring it to be scrapped.

During the cleaning process, some recipes introduce more particles into the chemical liquids than others. It is therefore distinguished between 'clean' and 'dirty' recipes (Gan et al. 2006). Depending on its configuration, a wet bench is dedicated to wafers which are associated with either 'clean' or 'dirty' recipes. Unless a wet bench is reconfigured, it can only be used to process wafers associated with either type of recipe.

Depending on the recipe mix, i.e., depending on the distribution of recipes in arriving wafer lots, the configuration of the WBTS has to be considered in order to maintain or improve the performance. Switching the settings of a particular wet bench from 'clean' to 'dirty' does not require any particular activity. However, when switching from 'dirty' to 'clean', the chemicals of the wet bench have to be changed. This process takes approximately three hours and requires a wet bench to be empty first. In any case, it is necessary to process all allocated wafer lots first, before switching to another configuration.

Arriving wafer lots are not directly allocated to a particular wet bench. They have to wait in a pending queue until a wet bench is available which can process the wafer according to its recipe without interruption. If the WBTS is properly configured, it is possible to process arriving wafer lots without causing congestion. However, if the recipe mix is changing, it is possible that the throughput of the WBTS will drop below the current load, causing the number of waiting lots in the pending queue to increase. The pending queue represents a buffer and, even when using an optimal WBTS configuration, there will be some fluctuation with periods of increasing number of pending wafer lots. This does not necessarily represent a problem.

If the number of pending wafer lots is steadily increasing it becomes obvious that the current configuration cannot handle the work load anymore. Increasing waiting times is reflected by an increasing cycle time. Semiconductor manufacturing companies are interested in keeping the cycle time low in order to avoid penalties due to missed deadlines. A steady increasing queue length is therefore considered as a critical condition. In order to detect a critical condition, we use a threshold for the queue length which is sufficiently high in order to avoid false triggering.

5.2 Symbiotic Simulation System

In a fully automated environment, decisions regarding the configuration of manufacturing equipment can be made by a symbiotic simulation control system (SSCS). The SSCS is responsible for performing the WIA process and to manipulate the physical system by using corresponding actuators. In the context of the WBTS the actuator is represented by the wet bench tool controller. This controller is responsible for draining and refilling the chemical liquids. Once the WIA process is triggered, various what-if scenarios are evaluated and a winner scenario is determined and its configuration implemented.

The SSCS observes the physical system using real-time sensor data. Depending on whether reactive WIA or preventive WIA is used, the WIA process is triggered differently. In case of reactive WIA, the WIA process is triggered if the current queue length exceeds a certain threshold (see Aydt et al. 2008). In case of preventive WIA, triggering is based on a periodically created forecast. In the context of our application, a forecast is created periodically every 12 hours and covers the anticipated performance of the physical system for the next 48 hours. If a critical condition is detected, i.e., if the queue length exceeds a certain threshold, the WIA process is triggered.

It is an objective to make decisions as quickly as possible. This is particularly true for reactive WIA. For performing experiments a cluster has been used which consists of 30 Dual core processors with 3.0 GHz and 4 GB of RAM. A WIA process needs approximately 1–6 minutes to finish

in this environment. This directly depends on the number of candidate action points which have to be analysed. For reactive WIA, only one action point is considered. It is therefore considerably faster than preventive WIA. At a typical wafer arrival rate of up to 1200 lots per day, the number of arriving lots per minute is less than one. A period of one minute is therefore considerably short and we can assume $t_a \approx t_{now}$ for reactive WIA. The forecast which is used by preventive WIA needs approximately 20 seconds.

In our application we consider a WBTS with eight wet benches of which each of them can be independently configured. Since a wet bench can be dedicated to either 'clean' or 'dirty' recipes, there is a total number of $2^8 = 256$ scenarios for each action point. We have chosen a simple method to effectively limit the number of candidate action points $t_{a,0}, t_{a,1}, ..., t_{a,k}$:

$$t_{a,i} = t_{a,i-1} - T * 2^{i-1} \tag{5}$$

for $i > 0$ and $t_{a,0} = t_c$ where T is a given time resolution. In our application context a time resolution of $T = 1$ hour has been used. The method is illustrated in Figure 4.

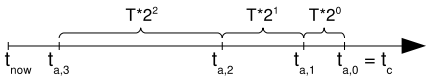

Figure 4: Selection of candidate action points $t_{a,k}$.

Although this method does effectively limit the number of candidate action points, it is not optimal. Development of a more sophisticated method is certainly justified but not further discussed here.

5.3 Error Handling Approach

We have chosen a simple error handling approach which involves using preventive WIA in combination with reactive WIA. The reactive triggering approach is only used as fallback solution to handle type I and type II errors.

If a critical condition is not detected in the forecast (type I error), no preventive WIA process is triggered. Although no preventive WIA process was triggered, the reactive WIA process is triggered immediately if the physical system becomes critical. Therefore, in the worst case, if the forecast always fails to detect a critical condition, there would be no advantage of using preventive WIA in comparison with reactive WIA. However, more importantly, there will be no disadvantage either because the performance of the physical system will never be worse as compared to the reactive approach. In case of a type I error, the preventive WIA will behave exactly like a purely reactively triggered symbiotic simulation system, i.e., $G_\alpha = 1$.

If a critical condition is detected even though this is not the case in the physical system (type II error), a WIA process is triggered and a solution for the alleged critical condition is determined and implemented. In a sense, this can be seen as a self-fulfilling prophecy because the symbiotic simulation system, which is expecting a critical condition, is actually causing it. Similarly to type I error handling, a reactive WIA approach can be used as fallback and is used to undo the changes caused by previous decision making, based on the preventive WIA process. It is possible that a wrong decision made by the preventive WIA can be undone by the reactive WIA fallback without causing a significant performance loss. However, in most cases it can be expected that the performance is significantly lower as compared to a system uses reactive WIA only, i.e., $G_\beta < 1$.

6 EVALUATION

For the evaluation of the symbiotic simulation system we use an emulator of the physical system rather than a real physical system for practical reasons. The emulator is realised as paced simulation and runs 7600 times faster than real-time. This allows us to cover time periods of several weeks when conducting experiments.

When comparing preventive WIA with reactive WIA, the specific G-values are important metrics to evaluate preventive WIA in the context of a particular application. These values are directly influenced by the used error handling approach. In our case, we have explained that G_α will always be equal to 1. Therefore, we will further focus on the empirical determination of the specific G-values for G_γ and G_β in this section.

6.1 Design of Experiments

Given that the load is sufficiently high, any recipe mix can cause the WBTS to become unstable. If this happens, the number of wafer lots in the pending queue is steadily increasing regardless the configuration. We already showed that a SSCS is capable of handling much higher loads than the common practise control approach (Aydt et al. 2008). Here, we consider only recipe mixes and loads for which there is at least one configuration that results in a stable system.

Before conducting the actual experiments, a number of random recipe mixes are chosen and the highest possible load is determined for which there is still at least one configuration that leads to a stable behaviour. We further divide this set of recipes mixes into two groups, consisting of recipe mixes that lead to stable behaviour for a 'major clean' or 'major dirty' configuration only, respectively. With a 'major clean' or 'major dirty' configuration we mean that the majority of the wet benches in a configuration are dedicated to wafer lots associated with either 'clean' or 'dirty' recipes, respectively

Pairs of recipe mixes are created by drawing randomly two recipe mixes, one from each group. These pairs are used to conduct experiment runs. For each run, the WBTS is initially configured with a stable configuration for the first recipe mix of the pair. After one week of simulation time, the recipe mix is changed to the second one of the pair. Since both recipe mixes have no configuration in common which would result in a stable behaviour, the control system is eventually forced to change the configuration of the WBTS.

A wafer lot generator is part of the simulation model which is used for both, the emulator and the what-if simulations. Change of recipe mixes are performed by instructing the lot generator accordingly. In order to analyse the behaviour for type II error situations, the error is artificially induced by preventing the lot generator of the emulator to change the recipe mix. However, the lot generators used for the what-if simulations and the forecasting simulation still performs the change. This results in a wrongly detected critical condition, i.e., a type II error.

Equation (1) can be used to calculate the G-values. For this purpose an assessment period has to be specified. In our experiments, we cause a change of recipe mix after 7 days. Therefore, we consider an assessment period starting at $t_s = 7$ days and ending when the experiment is over at $t_e = 14$ days. The cycle time factor, which is used as performance metric, is inversely proportional, i.e., the smaller the cycle time factor the better the performance. Therefore, it is necessary to transform the cycle time factor before calculating the G-value.

The functions $preventive(t_s,t_e)$ and $reactive(t_s,t_e)$ return the inverse of the average cycle time factor (denoted with $f_p(t_s,t_e)$ and $f_r(t_s,t_e)$) over the specified assessment period for preventive WIA and reactive WIA, respectively:

$$preventive(t_s,t_e) = \frac{1}{f_p(t_s,t_e)} \tag{6}$$

$$reactive(t_s,t_e) = \frac{1}{f_r(t_s,t_e)} \tag{7}$$

For example, let the mean cycle time factor for preventive WIA and reactive WIA over the assessment period be $f_p = 2.5$ and $f_r = 3.5$, respectively. This results in $preventive = \frac{1}{f_p} = 0.4$ and $reactive = \frac{1}{f_r} = 0.29$. The corresponding G-value is therefore $G = 1.38$.

6.2 Preventive WIA – Ideal Case

An example of the performance, using reactive and preventive WIA, for a major 'clean' to 'dirty' switch and vice versa is illustrated in Figure 5.

We performed various experiments, using a total of 10 different random pairs. In 9 and 8 out of these 10 cases,

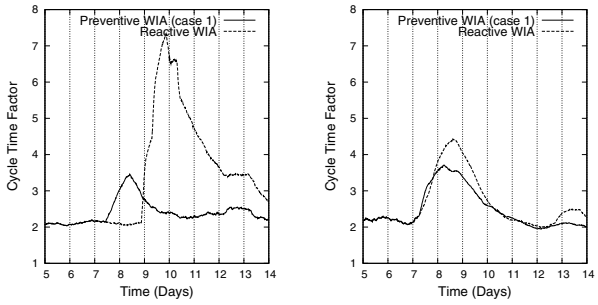

Figure 5: Cycle time factor of the WBTS when switching from a major 'clean' configuration to a major 'dirty' one (left) and vice versa (right) using preventive WIA and reactive WIA.

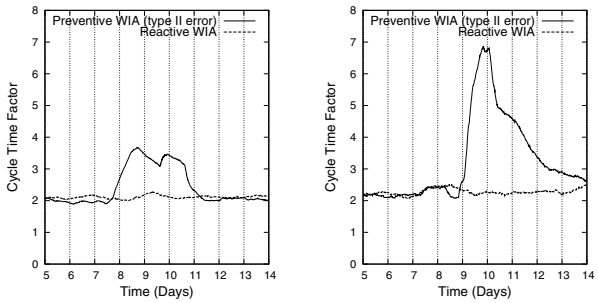

Figure 6: Cycle time factor of the WBTS when erroneously switching from a major 'clean' configuration to a major 'dirty' one (left) and vice versa (right) using preventive WIA in combination with reactive WIA.

the preventive WIA approach performed significantly better (with confidence level of 95%) when performing a major switch from 'clean' to 'dirty' and vice versa, respectively. The G_γ-values for the ideal preventive WIA case are illustrated in Table 2.

Table 2: Values for G_γ.

	Min G_γ	Max G_γ	Average G_γ
Clean to Dirty	1.03	1.51	1.21
Dirty to Clean	1.04	1.42	1.18

6.3 Preventive WIA – Type II Error

An example of the performance, using reactive and preventive WIA, under type II error conditions is illustrated in Figure 6.

Various experiments with a total of 19 different random pairs were conducted. When performing a major switch from 'clean' to 'dirty', the preventive WIA approach performed significantly worse in 18 out of these 19 cases. However,

when performing a major switch from 'dirty' to 'clean', the preventive WIA approach performed significantly worse in 16 and significantly better in 2 out of 19 cases. Analysis of these two cases has shown that even though the configuration has been changed erroneously (type II error), the final configuration, determined by the reactive WIA in order to handle the error, performed much better than the initial configuration used at the beginning of the experiment. This has caused the unexpected performance gain even under type II error conditions. In any case, a confidence level of 95% was used. The G_β-values for the type II error preventive WIA case are illustrated in Table 3.

Table 3: Values for G_β.

	Min G_β	Max G_β	Average G_β
Clean to Dirty	0.59	0.98	0.77
Dirty to Clean	0.59	1.25	0.85

6.4 Discussion of Results

Preventive WIA is capable of detecting a critical condition and can therefore reconfigure the physical system before the critical condition occurs. This is the major advantage of preventive WIA in comparison with reactive WIA where the symbiotic simulation systems reacts later, causing a more drastic performance drop (see Figure 5). In addition, since the performance drop is partly caused by the downtime of various wet benches, the performance gain of preventive WIA over reactive WIA is relatively low as compared to switching from a major 'clean' to a major 'dirty' configuration. An example is illustrated in Figure 7. It shows the performance gain of preventive WIA compared to reactive WIA.

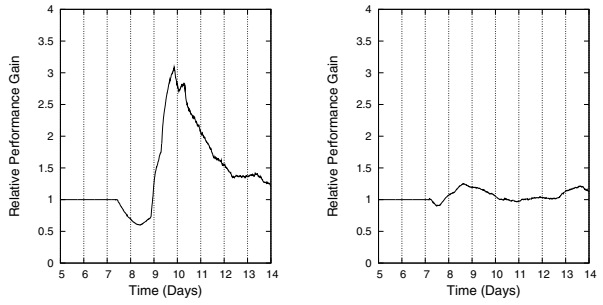

Figure 7: Relative performance gain of preventive WIA compared with reactive WIA when switching from a major 'clean' configuration to a major 'dirty' one (left) and vice versa (right).

In the ideal case, the WIA process makes the correct decision which results in G-values of $G_{\gamma,cd} = 1.21$ and $G_{\gamma,dc} = 1.18$ in average for a major switch from 'clean' to 'dirty' and vice versa, respectively. However, there is always the risk of an error. In any case, preventive WIA will be equally good as reactive WIA when dealing with type I errors, i.e., $G_\alpha = 1$. When dealing with type II errors, the average G-values are $G_{\beta,cd} = 0.77$ and $G_{\beta,dc} = 0.85$ for a major switch from 'clean' to 'dirty' and vice versa, respectively. The corresponding critical probability levels of $\varepsilon_{c,cd} = 0.52$ and $\varepsilon_{c,dc} = 0.45$ can be obtained by using (4). The G-values for the WBTS are illustrated in Figure 8.

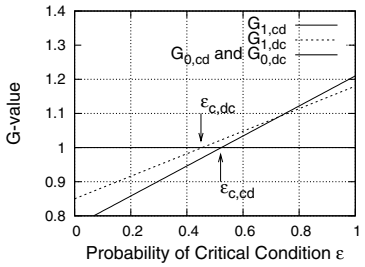

Figure 8: G-values for WBTS.

7 CONCLUSIONS

When applying preventive WIA, it is necessary to handle two different types of errors, which makes it less robust than reactive WIA. An appropriate error handling approach is required. In this paper we have proposed a simple error handling approach which uses reactive WIA as fallback solution. This approach is optimal for handling type I errors. However, a performance loss has to be taken into account when facing type II error situations. Although this approach is appropriate for the WBTS application, more sophisticated error handling approaches might be required in a different application context. For example, it might be possible to confirm previously detected critical conditions at a later time, before implementing preventive counter measures. This can be expected to reduce the number of unnecessary counter measures.

In practise it is necessary to make a decision, whether to use preventive WIA or not. We have introduced the notion of a G-value as metric which can be used for decision making if the error probability is known. In order for this method to be applicable, three specific G-values have to be determined (the fourth, G_δ, is always 1). We have described an example in the context of semiconductor manufacturing and explained how the G-values can be calculated. In our experiments, the preventive WIA illustrated superior performance for a majority of cases. The corresponding specific G-values G_γ and G_β have been determined empirically. By using these values we have determined the critical probability levels $\varepsilon_{c,cd} = 0.52$ and $\varepsilon_{c,dc} = 0.45$ which can be used for deciding whether to use preventive WIA or not in the context of the WBTS.

REFERENCES

Aloulou, M. A., and M.-C. Portmann. 2005. An efficient proactive reactive scheduling approach to hedge against shop floor disturbances. *Multidisciplinary Scheduling: Theory and Applications* 3:223–246.

Aydt, H., S. J. Turner, W. Cai, M. Y. H. Low, P. Lendermann, and B. P. Gan. 2008. Symbiotic simulation control in semiconductor manufacturing. In *Proceedings of the International Conference on Computational Science*.

Davis, W. 1998. On-line simulation: Need and evolving research requirements. In *Handbook of Simulation*, ed. J. Banks, 465–516. Wiley-Interscience.

Fujimoto, R., D. Lunceford, E. Page, and A. M. U. (editors). 2002, August. Grand challenges for modeling and simulation: Dagstuhl report. Technical Report 350, Schloss Dagstuhl. Seminar No 02351.

Gan, B. P., L. P. Chan, and S. J. Turner. 2006. Interoperating simulations of automatic material handling systems and manufacturing processes. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. Perrone, F. Wieland, J. Liu, B. Lawson, D. Nicol, and R. Fujimoto, 1129–1135: IEEE Press Piscataway, NJ, USA.

Gan, B.-P., P. Lendermann, K. P. T. Quek, B. van der Heijden, C. C. Chin, and C. Y. Koh. 2006. Simulation analysis on the impact of furnace batch size increase in a deposition loop. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. Perrone, F. Wieland, J. Liu, B. Lawson, D. Nicol, and R. Fujimoto, 1821–1828: IEEE Press Piscataway, NJ, USA.

Kamrani, F., and R. Ayani. 2007, October. Using on-line simulation for adaptive path planning of UAVs. In *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-time Applications*, 167–174. Chania, Greece.

Low, M. Y. H., K. W. Lye, P. Lendermann, S. J. Turner, R. T. W. Chim, and S. H. Leo. 2005. An agent-based approach for managing symbiotic simulation of semiconductor assembly and test operation. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, 85–92. New York, NY, USA: ACM Press.

Low, M. Y. H., S. J. Turner, D. Ling, H. L. Peng, L. P. Chai, P. Lendermann, and S. Buckley. 2007. Symbiotic simulation for business process re-engineering in high-tech manufacturing and service networks. In *Proceedings of the 2007 Winter Simulation Conference*, ed. S. Henderson, B. Biller, M. Hsieh, J. Shortle, J. Tew, and R. Barton: IEEE Press Piscataway, NJ, USA.

Pfund, M. E., S. J. Mason, and J. W. Fowler. 2006. Handbook of production scheduling. In *Semiconductor Manufacturing Scheduling and Dispatching*, ed. J. W. Herrmann, Volume 89 of *International Series in Operations Research & Management Science*, 213–241. Springer New York.

Scholl, W., and J. Domaschke. 2000, August. Implementation of modeling and simulation in semiconductor wafer fabrication with time constraints between wet etch and furnace operations. *IEEE Transactions on Semiconductor Manufacturing* 13 (3): 273–277.

AUTHOR BIOGRAPHIES

HEIKO AYDT is a Research Associate and part-time PhD student in the School of Computer Engineering at Nanyang Technological University (NTU) in Singapore. He received his M.Sc. from the Royal Institute of Technology (KTH) in Stockholm. His e-mail address is <aydt@ntu.edu.sg>.

STEPHEN JOHN TURNER is Professor in the School of Computer Engineering and Director of PDCC at NTU. He received his PhD in Computer Science from Manchester University (UK). His e-mail address is <assjturner@ntu.edu.sg>.

WENTONG CAI is an Associate Professor in the School of Computer Engineering at NTU. He received his PhD in Computer Science from University of Exeter (UK). His e-mail address is <aswtcai@ntu.edu.sg>.

MALCOLM YOKE HEAN LOW is an Assistant Professor in the School of Computer Engineering at NTU. He received his D.Phil. in Computer Science from Oxford University (UK). His e-mail address is <yhlow@ntu.edu.sg>.

PETER LENDERMANN is the Co-Founder and CEO of D-SIMLAB Technologies. He received his PhD in Applied High-Energy Physics from Humboldt-University and an MBA in International Economics and Management from SDA Bocconi. His e-mail address is <peter@d-simlab.com>.

BOON PING GAN is the Founder and CTO of D-SIMLAB Technologies. He received his Master of Applied Science in Computer Engineering from NTU. His e-mail address is <boonping@d-simlab.com>.

RASSUL AYANI is Professor in the School of Information and Communication Technology (ICT) at KTH. He is also visiting Professor in the School of Computer Engineering at NTU. He received his PhD from KTH in Stockholm. His e-mail address is <ayani@kth.se>.