Advances in Simulation for
Production and Logistics Applications
Markus Rabe (ed.)
Stuttgart, Fraunhofer IRB Verlag 2008

Training Simulator for Rolling Mill Maintenance

Trainings-Simulator für die Instandhaltung von Walzstraßen

Gerhard Rath, University of Leoben (Austria)
Johannes Zorn, Erich Könighofer,
voestalpine Schienen GmbH, Leoben Donawitz (Austria)

Abstract: The development of a real-time simulation for operator training is presented. The operation of a rolling mill incorporates the procedure of the roll exchange, which is performed autonomously under the control of the automation system. In exception situations, the maintenance personnel have to finish the complex procedure. To enhance the training skills independently from the real machine, a training simulator was developed. The structure of the hardware and software system is presented, and a way to estimate the development effort is given. The system proved its suitability for training in practice and brought remarkable benefits for the rolling mill maintenance.

1 Introduction

1.1 The Ultra Flexible Reversing (UFR) Mill

The voestalpine Schienen GmbH is a leading manufacturer of rails and is running a new ultra flexible reversing mill (UFR, fig. 1) consisting of three identical mill stands (Pfeiler et al. 2003). The automation system of the UFR mill provides level 1 and level 2 automation, including sequential control of the movements of the mechanical equipment, material tracking and data logging, minimum tension control and closed loop control of the hydraulic and electric drive systems.

1.2 Maintenance and Roll Exchange

In normal operation, the complex roll exchange process runs completely autonomously under the supervisory control system. With the help of hydraulic robots, the mill stands are shifted and then split into two halves, which leaves the baskets (packets with rolls and guides), on a carriage for fast exchange. In this automatic mode the operation continues many days around the clock without problems. When a failure situation occurs, the automation system stops and leaves the control to the

***Figure 1**: The UFR mill at voestalpine Schienen GmbH*

maintenance staff. After fixing the problems, they have to operate dozens of drives manually until the automation program can pick up the process control again. Here the long period of trouble-free automated production, which is of course a desirable advantage of the whole plant, turns out to cause a problem for the maintenance personnel. They have no opportunity to train their skills during the routine production. Since the roll exchange procedure is highly complex and the electronic control is less strict in the manual mode, any mistake can cause a severe damage. The contradictory demands of uninterrupted production and sufficient training opportunities led to the application of a machine simulation that replaces the real plant (Hardware-in-the-loop, Isermann et al. 1999; Ledin 1999; Kabitzsch et al. 2006).

2 Training Simulator for the Roll Exchange

2.1 Requirements of the Training Simulator

The aim of the development is to enable training of the maintenance staff independently from the real rolling mill.

The diagram in figure 2 shows the use cases (Pilone and Pitman 2005, p. 77) associated with the roll exchange. During routine production, the machine operator supervises the machine and its control programs. The roll exchange is performed automatically and requires no human interaction. After a stop due to a problem, the maintenance personnel have to carry out manual operation steps. A simulator that enables training of these operations under almost real conditions must provide an identical ‘copy’ of the use case.

The maintenance person operates the roll exchange system via the programmable logic control (PLC) and its human-machine interface (HMI). Since the trainee should not see a difference to reality, the HMI and the PLC must be identical copies of the plant. From the view of the control system (PLC) the simulator has to replace the real plant and to provide an identical behaviour.

The most important consequence is that the simulation software must be able to emulate the behaviour of the machine in real-time (Kopetz 1997, p. 266).

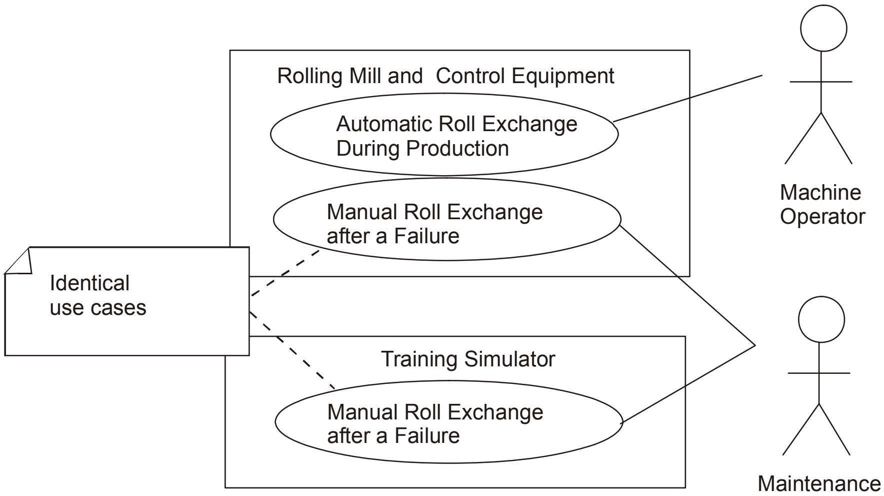

***Figure 2**: Use cases of the roll exchange procedure*

Of course not only functional, logical and ergonomic aspects are to be fulfilled, but also costs must be taken in account. Several hundreds of inputs and outputs are communicated between PLC and the simulation. In the real plant, most of the I/Os are wired in control cabinets, which is very expensive. Instead of hard-wired connections, it is preferable to have a fast field-bus line that has enough capacity to transfer all the information in real-time (Kweon et al. 1999).

2.2 Structure of the Training Simulator

The requirements presented in the previous section led to the structure shown in figure 3. As software platform WINMOD was chosen. This system runs on an ordinary PC fast enough to have real-time conditions. Communication boards in the PC establish PROFIBUS connections to the logic control system of the type S7 from Siemens.

The simulator, which is running on a PC, replaces the rolling mill (fig. 3). An identical control system, its visualisation (WinCC from Siemens) and the operating panel (HMI) are used for the training simulator. Consequently, the trainee meets the same operation conditions as in reality. Furthermore, the software of the PLC, the visualisation and the operating panel need not to be changed.

2.3 Software Design

The most important issue throughout the development process was to reduce the complexity to a reasonable extent (Lieberman 2003). At first the basic process of a training session was designed, figure 4 shows the states of such a session.

Of course, the simulation has to run also through the automatic sequence like during routine production. This is not a training condition, but a good testing milestone for the software. After a (simulated) failure the actual training starts ('manual control' in fig. 4). When the operator makes a severe mistake, the training is over. This eliminates the need to model repairs or other recovery actions.

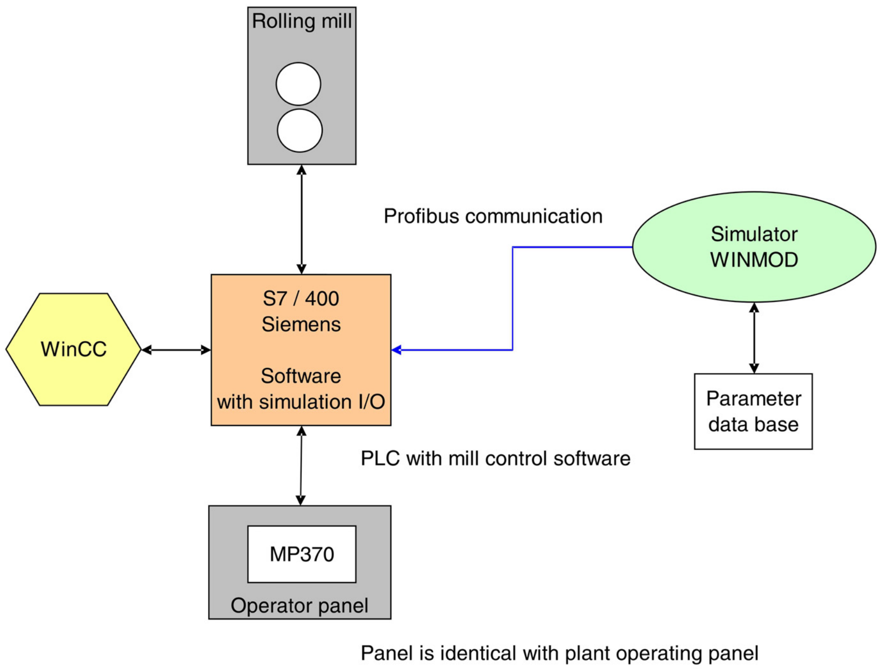

Figure 3: *Structure of the training simulator*

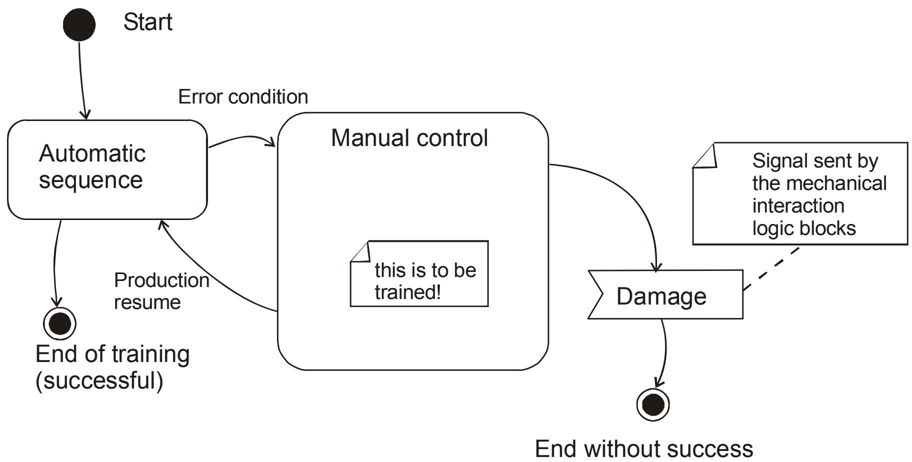

Figure 4: *State diagram for the training session*

The artifacts of the simulation software system that were developed to fulfil the requirements are displayed in the class diagram in figure 5.

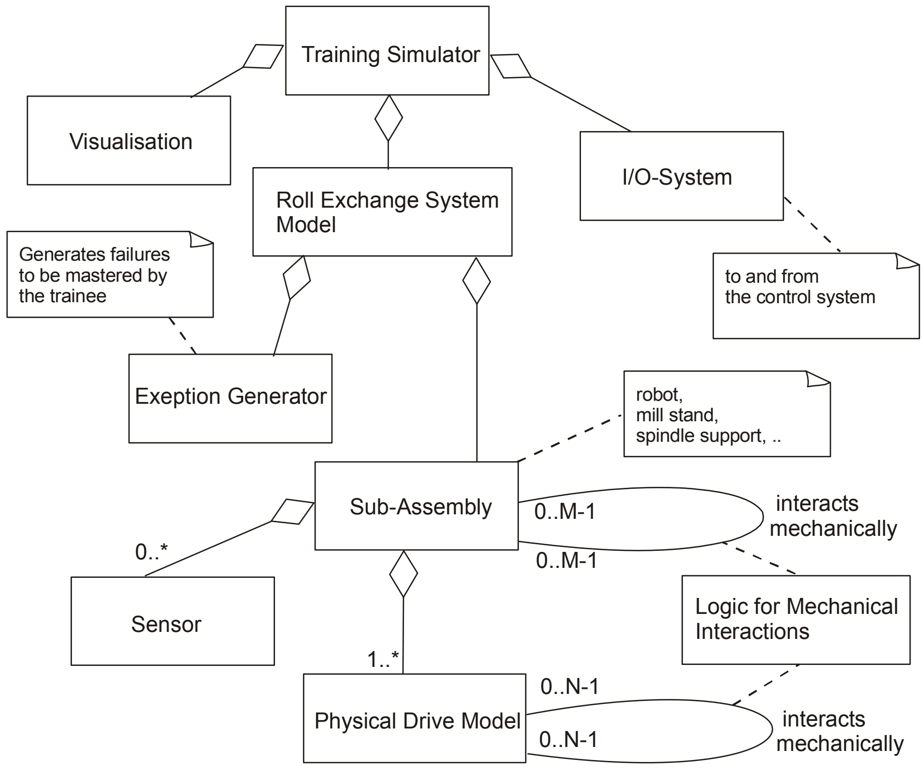

Figure 5*: Class diagram for the training simulator*

The 'molecules' of the whole system are the physical drive models (fig. 5) and are modelled according to the basic methods of system dynamics (Rowell and Wormley 1997). They drive parts or autonomous sub-assemblies of the rolling mill, for example the robot, the mill stand or a conveyor table. Drives as well as sub-assemblies can have mutual interactions, for example, mechanical parts can block each other. Some interactions can cause a damage which is a severe mistake caused by the trainee. These situations lead to the end of the training session. Each interaction must be modelled by a logic block.

Another useful view of the modelling system are the states of a drive that are assumed during a training session (fig. 6). The 'normal motion' is the basic and easiest task of modelling. Exception situations are divided into two categories:

1. Collisions are situations where systems block each other but cause no damage, consequently they can be resolved by the trainee and the process may continue.

2. A damage is a severe problem that requires repair or other recovery actions. The trainee is informed about this mistake by terminating the session.

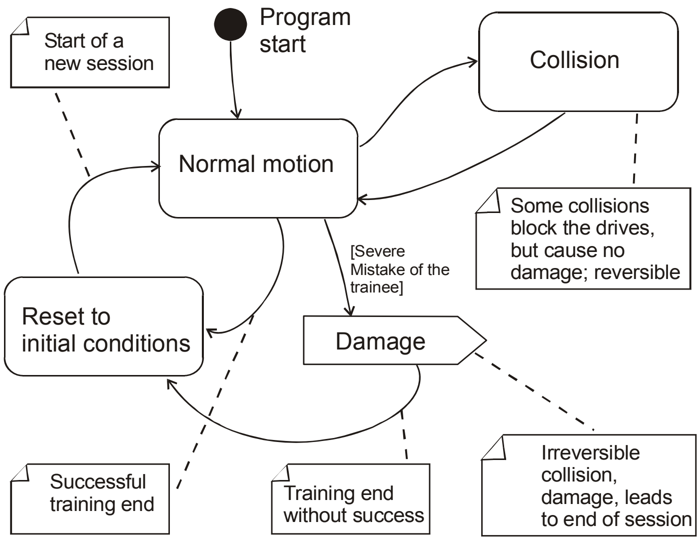

Figure 6*: States of a single drive unit during a training cycle*

The stop of the session saves us from modelling this kind of exception. Unfortunately it is required now to bring the system into a well-defined situation ('reset') before one can start a new session again. Like in reality, it is not possible to do this simply by initialising some variables, since the PLC system does not accept arbitrary changes without causing errors. Restarting the control system is much too complex and is to be avoided. So the 'reset' of the model is a complex combination of setting state variables, doing some motions and initialising some spots in the PLC program.

2.4 Estimating the Amount of Programming Work

Looking back to figure 5, one can see that drives (cylinder, electrical motors or other motion units) or mechanical sub-assemblies of the rolling mill can interact with each other. Since the number of interactions can be very large, they are an essential factor for the complexity of the system and for the developing effort. Consequently, a good estimation for this number was demanded before starting the design work.

The worst case (largest number of interactions) is obtained, when every unit can interact with any other.

$$L_{\max} = \binom{N}{2} + \binom{M}{2} \qquad (1)$$

with:

N ... Number of drive units in the model
M ... Number of mechanical sub-assemblies in the model
L ... Number of possible interactions
L_{max} ... Upper estimate for the number of interactions

Assuming N=40 and M=8 for the actual rolling mill model, the maximum of possible interactions will be:

$$L_{\max} = 808 \tag{2}$$

Equation (1) is the worst case for such a system. Fortunately it turns out, that manipulation systems of heavy machinery plants have a certain structure that leads to an essential reduction of the amount of interactions. A drive usually has two limit positions where an interaction with a neighbouring drive can occur. Such manipulation systems tend to be a linear chain of drives or subsystems (fig. 7). For example, a load is handed over at the end point of the first drive to the subsequent drive.

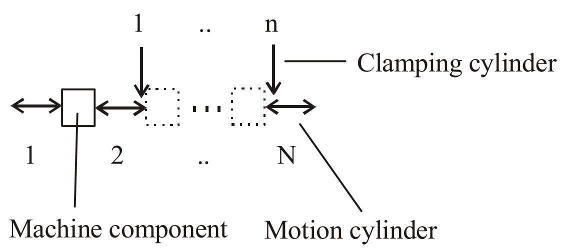

***Figure** 7: Linear chain of motion cylinders*

Similar are the interactions between complex sub-assemblies. If N is the number of drives and M the number of sub-assemblies, the number L of interactions for a linear chain is:

$$L = (N-1)+(M-1) = N+M-2 \tag{3}$$

Some of the drives are clamping, locking or tooling cylinders which have the possibility of an interaction only at one end of their motion path. Similarly some sub-assemblies, for example the screwing robot cannot collide with anything in their off-position. If n is the number of drives and m the number of sub-assemblies with only one possible interaction, the total of interactions does not change compared to equ. (3):

$$L = \left[(N-n)-1+n\right]+\left[(M-m)-1+m\right] = N-1+M-1 \tag{4}$$

With these assumptions, the lower limit for the estimated mechanical interactions is:

$$L_{\min} = N+M-2 \tag{5}$$

For the current example with N=40 and M=8 the estimate is much lower compared to the worst case in equ. (2):

$$L_{\min} = 46 \tag{6}$$

Additional reduction of the programming effort comes from the nature of interactions (fig. 8). A number of potentially dangerous actions are prohibited by the program of the electronic control and need not to be regarded in the simulation model. Furthermore, a certain number of interactions are unknown or ignored, since they are extremely unexpected.

From the practical experience during the programming work it turned out, that the actual number of logic blocks to be programmed is quite close to the estimate from equ. (6).

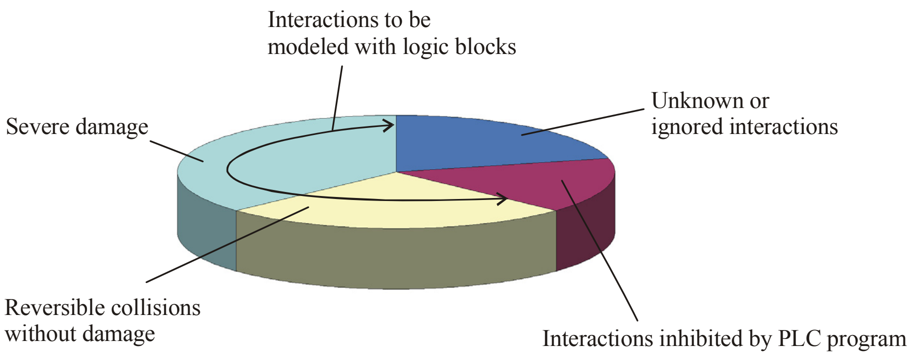

***Figure 8**: Classifications of interactions*

3 Practical Training Experience

Regular training sessions were carried out using the simulator equipment. A trainer supervised the trainees who had to fulfil certain tasks on the simulator. The acceptance of the system was very good. After the training, the participants reported they had learned new aspects about the rolling mill system and had increased their assurance in handling the apparatus.

After a test period of some months, the following benefits turned out:

1. Training sessions with the simulator instead of the real machine reduce the risk of damage to the real components in the case of a mistake.
2. More training can take place without stopping the production. Only periodical learning prevents forgetting of abilities.
3. Trainings can take place at convenient times, e.g. stand-by times, and need not happen during a production stop maybe even in Saturday night.
4. The training program is standardised and contains all exercises that are essential. In reality seldom all functions are operable at a certain training time.
5. Well-trained operators act faster and more accurate. The restart of production needs less time, and the risk of damage due to mistakes is reduced.

6. Sometimes the cause of an error in the complex technical system stays unclear. On the simulator critical machine situations can be repeated virtually. This helps to find explanations of such unclear errors.
7. New components of the control software can be tested on the simulator before their implementation on the real machine. This decreases the down-time of the plant and reduces risk essentially.

All advantages that are mentioned previously were presumed and intended before launching the project. Additionally new unexpected experience could be made:

8. First of all, the engineer gets a better understanding about the control programs that were made years ago by external staff. This is important to understand the functions of the controller, especially if extensions are planned.
9. Furthermore even severe bugs in the actual control program were found, which fortunately did not have an effect up to the present.
10. Finally, mistakes in the HMI, the operator panel, could be detected, for example wrong labels or swapped signals.

4 Conclusions

A Hardware-In-The-Loop simulation was applied successfully to train the roll exchange procedure in a tandem rolling mill. The acceptance by the maintenance staff was excellent. The system turned out to have great advantages: More time at free scheduling allows periodical training sessions, the risk of damage for the mill is reduced, and restart times after a problem are shorter. As additional positive side effects, the simulator helps to identify unclear failure sources, and can be used as a test stand for new or changed automation software components.

5 Further Research

Today the technology to create real-time simulators for complex systems is available. Further development is to be done on a didactical level to improve the training effect (Sauvé et al. 2007). At present, sessions without the supervision of a trainer are carried out which will be evaluated for further developments.

References

Isermann R.; Schaffnit J.; Sinsel S. (1999) Hardware-in-the-loop simulation for the design and testing of engine-control systems. Control Engineering Practice 7 (1999) 5, pp. 643-653

Kabitzsch, K.; Vasyutynskyy, V.; Theiss, S. (2006) Monitoring und Ergebnisauswertung für Hardware-in-the-Loop-Simulationen. In: Wenzel, S. (Hrsg.): Simulation in Produktion und Logistik 2006, Tagungsband zur 12. Fachtagung Simulation in Produktion und Logistik, Kassel. Erlangen: SCS Publishing House, pp. 351-360

Kopetz, H. (1997) Real-time Systems – Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, Dordrecht

Kweon, S.K.; Shin, K.G.; Zheng, Q. (1999) Statistical real-time communication over ethernet for manufacturing automation systems. Proceedings of the 5th IEEE Real-Time Technology and Applications Symposium, Vancouver, Canada, pp. 192-202

Ledin, J.A. (1999) Hardware in the Loop Simulation, Embedded Systems Programming 12 (1999) 2, pp. 42-53

Lieberman, B.A. (2003) The art of modeling. The Rational Library, http://www-128.ibm.com/developerworks/rational/library/2741.html, 07.07.2008

Pfeiler, H.; Köck, N.; Schröder, J.; Maestrutti, L. (2003) The new rail mill of voestalpine Schienen at Donawitz. MPT International 26 (2003) 6, pp. 40-44

Pilone, D.; Pitman, N. (2005) UML 2.0 in a Nutshell, O'Reilly, Köln

Rowell, D.; Wormley, D.N. (1997) System Dynamics: An introduction, Prentice-Hall, Upper Saddle River

Sauvé, L.; Renaud, L.; Kaufman, D.; Marquis, J.S. (2007) Distinguishing between games and simulations: A systematic review. Educational Technology & Society, 10 (2007) 3, pp. 247-256