SSM Failure models

Ivan S Zapreev

The modifications table

Date	Description
09.06.2004	Initial version with two models “Simple Failure model” and “Recovery model”.
10.06.2004	Added two new models. One for fixed number of cards and one for nondeterministic number of cards.
11.06.2004	Added few models (nondeterministic number of cards to fail with one station and a DTMC model) added all the rest models to the appendix
14.06.2004	Added Fixed cards model for one station. Minor fixes.
09.07.2004	Added model with the Weibull based distribution of failures.

Table of contents

1. Common description 3
2. Simple Failure model 3
   a. Model description 3
   b. Modeling results 3
   c. Common formula derivation 4
3. Recovery model 6
   a. Model description 6
   b. Modeling results 6
   c. Common formula derivation 7
4. Fixed cards model for 4 stations and the schedule 10
   a. Model description 10
   b. Modeling results 10
5. Nondeterministic cards model for 4 stations 12
   a. Model description 12
   b. Modeling results 12
6. Nondeterministic cards model for one station 13
   a. Model description 13
   b. Modeling results 13
7. DTMC model for one station 16
   a. Model description 16
   b. Modeling results 17
8. Fixed cards model for one station 18
   a. Model description 18
   b. Modeling results 18
9. Weibull based model for one station 19
   a. Model description 19
   b. Modeling results 20
10. Conclusions 21
11. Future work 21

12. Appendix A. 21
13. Appendix B. 22
14. Appendix C. 24
15. Appendix D. 25
16. Appendix E. 27
17. Appendix F. 27
18. Appendix G. 28
19. Appendix H. 28

1. Common description

Here we describe several models for the Cybernetix Case Study.

Originally the common basis for models is that there are 4 personalization stations and one load and unload station. System also contains a belt that can move only from left to right and which transfer personalizing cards.

There are the following limitations:

- 10 time units are required to personalize card in a personalization station.
- 1 time unit is required to move the belt.
- 2 time units are required to load and unload card and while loading and/or unloading the belt can not move.
- Getting and putting cards by personalization stations do not take time.
- Card can stay in personalization station longer then the time required to personalize it.

The algorithm (further schedule) called Super Single Mode (SSM) for 4 personalization stations rules the whole system.

Our main interest is focused on involving probabilistic failures of personalization stations into this model and gathering probabilities for the number of broken cards.

In other words the main question here is:

If the system processes N cards then what is the probability that after the work is done the number of broken cards is equal to the certain value M?

Below we will introduce several personalization station failure models and will present results gathered by the model checking of these models and also some formulas that have been derived. Some of these models are very close to the description of the system presented above and some of them are very abstract but still make sense as allow calculating required probabilities easily.

2. Simple Failure model

In this section we will present the model, which can be described as a model where cards can be broken by the personalization station with the certain probability. In other words personalization station is not crashing itself but can damage cards.

a. Model description

The main idea of this model is that we involve broken cards. The order of broken cards in not relevant and so the whole schedule remains to be the SSM.

Each card after personalization is put to the belt and we assume that with probability p it is not broken. We also assume that the value of p is 0,999 and thus for each card the probability to be broken after personalization is equal to 0,001.

b. Modeling results

The corresponding model has been created for the PRISM tool and was used with the SSM schedule for 12 cards. The resulting plot of gathered probabilities is present below (Figure 1.).

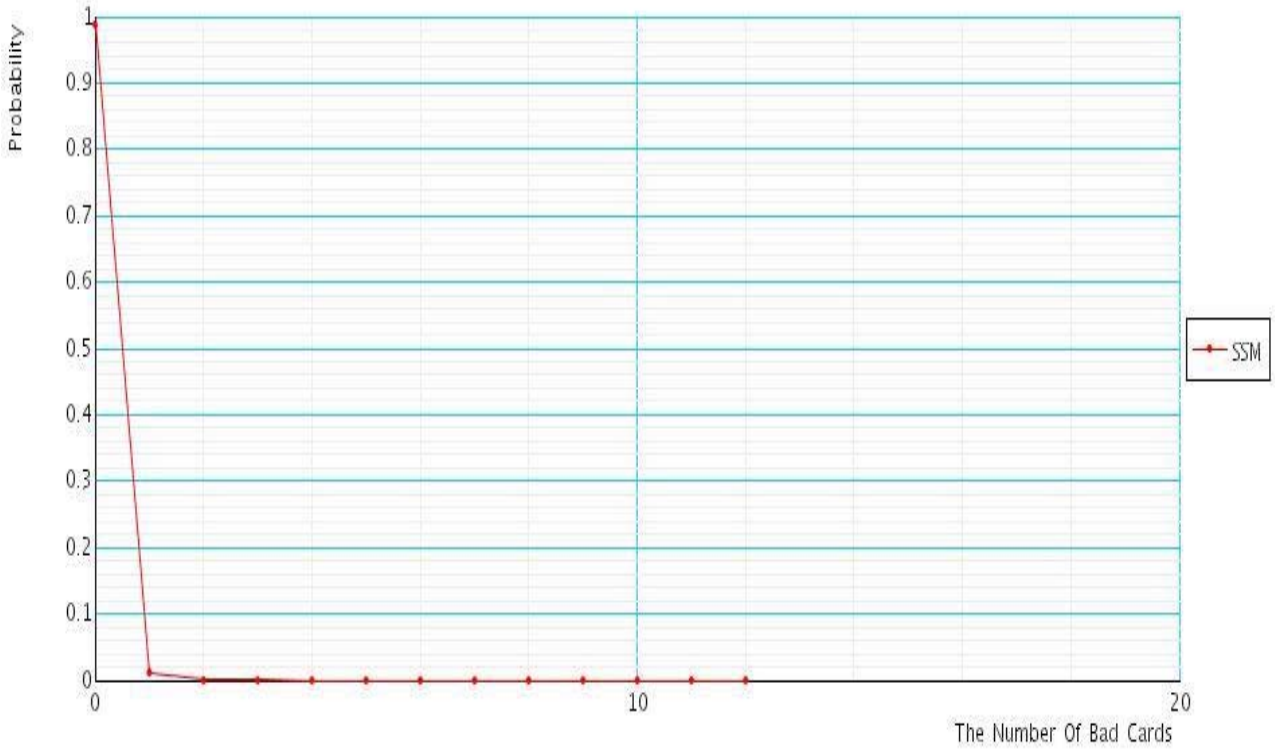

Figure 1. The probability that a certain number of 12 cards will be broken

The corresponding values are represented in the Table 1 below.

M	$P(M\ of\ 12\ cards\ are\ broken)$
0	0.988065780494209
1	0.01186865802395446
2	6.534296209384338E-5
3	2.1802790154769228E-7
4	4.910538323146222E-10
5	7.864726043077035E-13
6	9.184698415338544E-16
7	7.880479120839591E-19
8	4.93022968020495E-22
9	2.1934065978000004E-25
10	6.586806600000001E-29
11	1.1988000000000004E-32
12	1.0000000000000004E-36

Table 1. Values of the function present on the Figure 1

The used PRISM model without schedule is presented in the Appendix A.
Now let us try to derive the common probability formula for this model.

c. Common formula derivation

It is obvious that if p is the probability for a card not to be broken then

$$P(M'th\ card\ has\ been\ broken \mid All\ \ previous\ cards\ has\ not\ been\ broken) = p^{N-1}(1-p)$$

As far as a personalized card can be marked as broken only while it is put to the belt by a personalization station. This process is independent from other personalization stations and

cards. Hence the probability that M of N cards will be broken by K personalization stations that are present in the system is the following:

$$\prod_{i=1}^{K} p^{N_i - M_i}(1-p)^{M_i} = p^{N-M}(1-p)^{M}$$

$$\text{where } \sum_{i=1}^{K} M_i = M \text{ and } \sum_{i=1}^{K} N_i = N .$$

Note that this formula is true for fixed M_i and N_i values, where M_i is the expected number of broken cards produced by the $i'th$ station that totally processed N_i cards. If we also take into account the number of possible combinations then it is obvious that the probability that M of N cards will be broken does not depend from the number of stations and schedule and is the following:

$$P(M \ of \ N \ cards \ are \ broken) = C_N^M \, p^{N-M}(1-p)^{M}$$

$$\text{where } C_N^M = \frac{N!}{M!(N-M)!}.$$

This formula gives the same results as the ones gathered by the PRISM tool for the Simple Failure model.

Let's now check the behavior of the function $P(1 \ of \ N \ cards \ is \ broken)$ for different values of N. In this case we have the following simple formula to calculate probabilities:

$$P(1 \ of \ N \ cards \ is \ broken) = Np^{N-1}(1-p)$$

The plot of this function can be seen on the Figure 2.

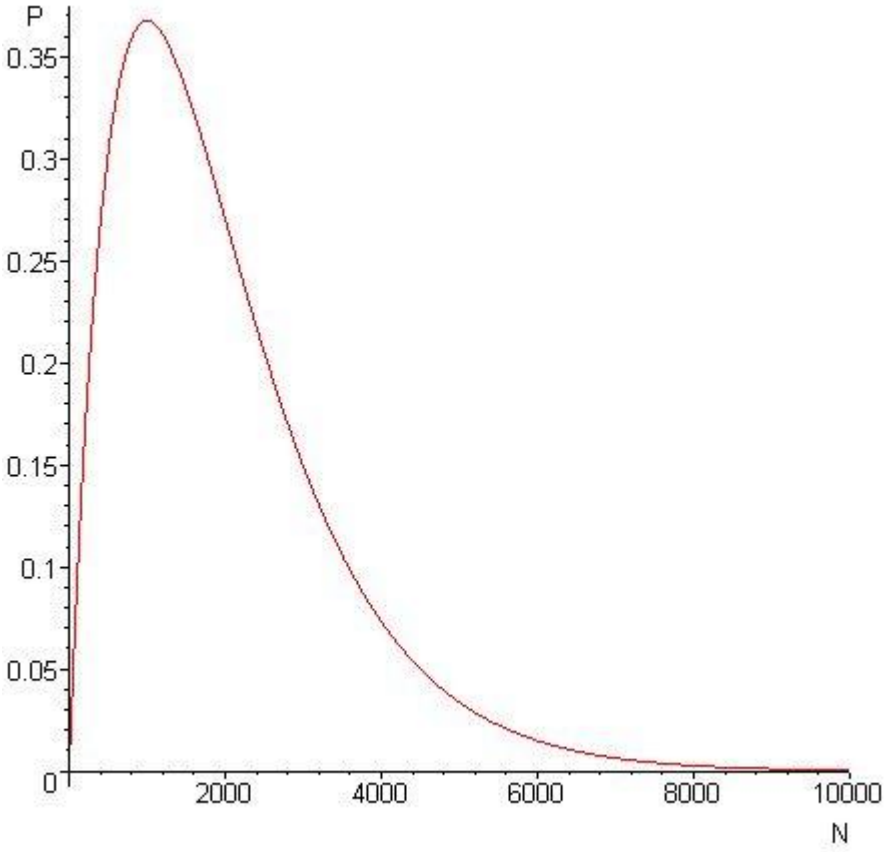

Figure 2. The $P(1 \ of \ N \ cards \ is \ broken)$ function values

It is obvious from the curve that is present on the Figure 2 that the probability that one card is broken is first increasing up to certain value and then is decreasing asymptotically approaching zero. This behavior is explained by the fact that although intuitively evident that the more cards are processed the higher should be the probability of that at least one card will be broken at the

same time as far as we deal with the increasing number of possibilities for the number of broken cards the probability of each one of them is getting lower.

3. Recovery model

In this section we will present the model with failures in which personalization station crashes and needs a constant time to recover. During this time period it produces broken cards.

a. Model description

The main limitations of this model are the following:

- Personalization station can crash only while it is working or just holding a card and is not broken,
- Personalization station is recovering even if it doesn't work,
- It requires a certain time to be repaired (25 time units for our PRISM model),
- When broken a personalization station continues to personalize cards,
- If a station was broken while personalizing a card then the card is marked as broken.

b. Modeling results

The corresponding model has been created for the PRISM tool and was used with the SSM schedules for 12 cards. The resulting plot of gathered probabilities is present below (Figure 3.).

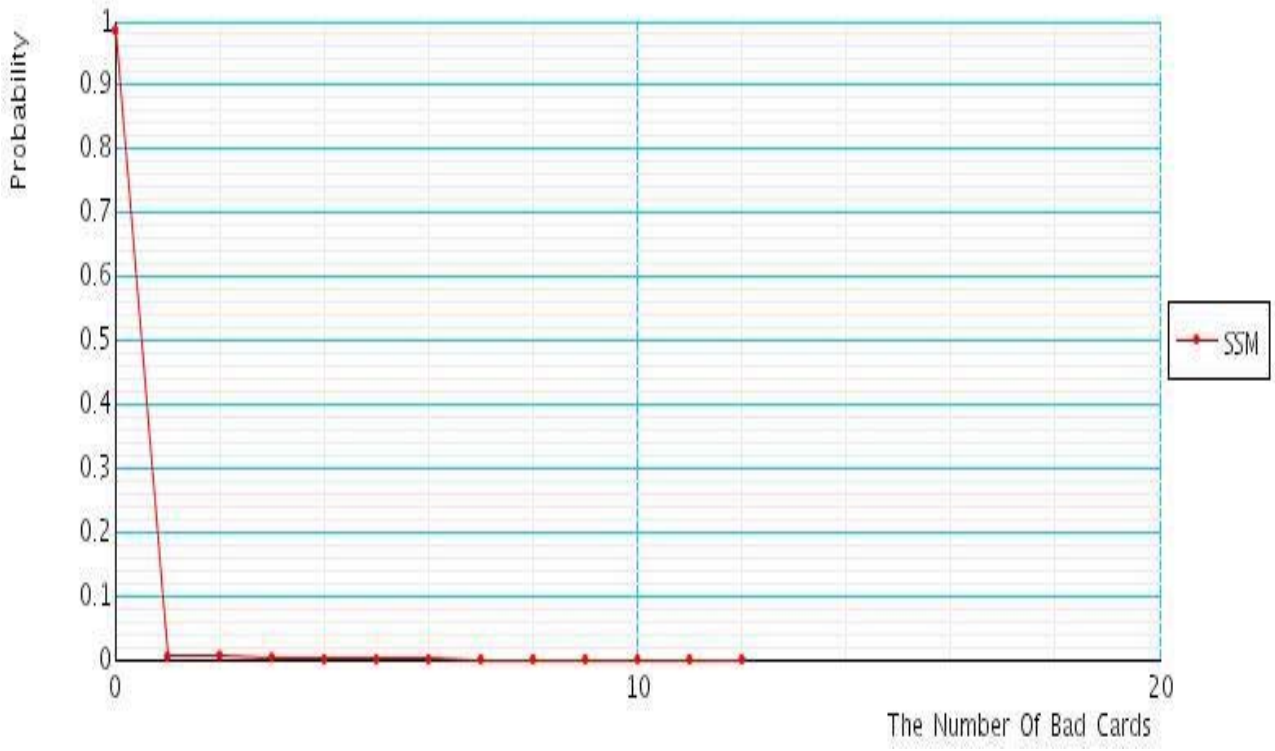

Figure 3. The probability that a certain number of 12 cards will be broken

The corresponding values are presented in the Table 2. below.

M	$P(M\ of\ 12\ cards\ are\ broken)$
0	0.9864913913197956
1	0.004343187341540462
2	0.006332348037085769
3	0.002792335594596362
4	2.4362908506248572E-5
5	1.3370905148151182E-5
6	2.965302750784101E-6
7	2.7796552271929267E-8
8	9.393831609268534E-9
9	1.388770397019918E-9
10	8.996876932987969E-12
11	2.191359508694111E-12
12	2.400380861370896E-13

Table 2. Values of the function present on the Figure 3

The local maximum for two cards ($M = 2$) can be explained by the length of the recover period of the personalization station as 25 time units length repairing period almost each time covers two working periods of the personalization station except if station crashes during processing of the last card.

The used PRISM model without schedule is represented in the Appendix B.

Now let us try to derive the common probability formula for this model.

c. Common formula derivation

For this model it goes without saying that probability of the certain cards to be broken depends on the used schedule. It is also quite obvious that SSM mode should have a working cycle for each personalization station so the algorithm can be stable. I.e. there should be a periodical sequence of working and suspending periods of certain length.

As far as we have SSM for 4 personalization stations it was experimentally checked that there is the same working cycle for each of four personalization stations. It is a sequence of constant suspending and working periods i.e. (SW) where S is the time while personalization station does not work (suspending period) and W is the time while personalization station works and holds a card.

It is quite obvious that this cycle doesn't hold for the beginning of work and for the end of work i.e. when we process the first cards and the last cards but if the number of cards is big then most of them are processed by this working cycle. Experimentally it was found that $S = 3$ and $W = 12$.

Let us first consider a single personalization station with the working cycle and then extend this case for 4 personalization stations.

Single personalization station

Let $M(W, S, R, t)$ be the number of cards that will be broken in the situation when station crashes (Figure 4.). Here R is time needed for repairing, S is the suspending time of the personalization station, W is the working time of the personalization station and t is the number of time units before the end of current working period of the personalization station.

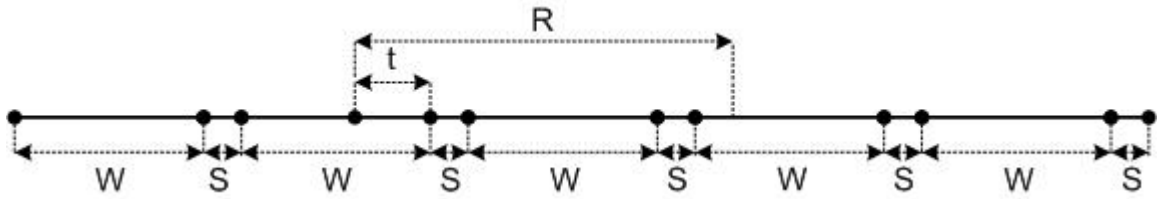

Figure 4.

then $M(W,S,R,t)$ can be calculated by the following formula:

$$M(W,S,R,t) = 1 + \left[\frac{R-(t+S)}{W+S}\right] + \delta(W,S,R,t)$$

where $[\ ]$ is the integer part of the result, and $\delta(W,S,R,t) = \begin{cases} 1, & \langle (R-(t+S))/(W+S) \rangle > 0 \\ 0, & else \end{cases}$

where $\langle\ \rangle$ is a fractional part of the number including a sign of the number.

If S_N is the number of broken cards produced by one personalization station then for N cards

$$P(S_N \le 0) = p^{WN}$$
$$P(S_N \le L) = ? \text{ where } L \in 1..N-1$$
$$P(S_N \le N) = 1$$

It is easy to see that the first parity can be strengthened in the following way:

$$P(S_N \le M_{\min}) = P(S_N \le 0) = p^{WN} \text{ where } M_{\min} = M(W,S,R,W)$$

Now let us define a set of equations and inequalities that can be used to calculate the value of $P(S_N \le L)$ when $L \in 1..N-1$. To do this first let us change the formulas in order to use the absolute time instead of t.

Let *time* be the absolute time value i.e. time passes from the beginning of system working then:

$$t = t(time) = \begin{cases} W - time\%(W+S) + 1, & if\ 1 \le time\%(W+S) \le W \\ 0, & else \end{cases}$$

Here % is a residue of division and inequality is used to state that a station can break only while it is working.

Now we have $M_{time}(W,S,R) = M(W,S,R,t(time))$.

If station has crashed at time $time_1$ then the next time it can crash is $time_2$ and $time_1 + R \le time_2$. We should also note that if

$$\left[\frac{time_1 + R}{W+S}\right] = \left[\frac{time_2}{W+S}\right]$$

then

$$M_{time_2} = M_{time_2}(W,S,R) - 1$$

Which means that in one working period personalization was broken at the beginning and after it had been repaired it crashed again but as far as the card that was processed had been already broken the number of broken cards for the next period is 1 card less (Figure 5.).

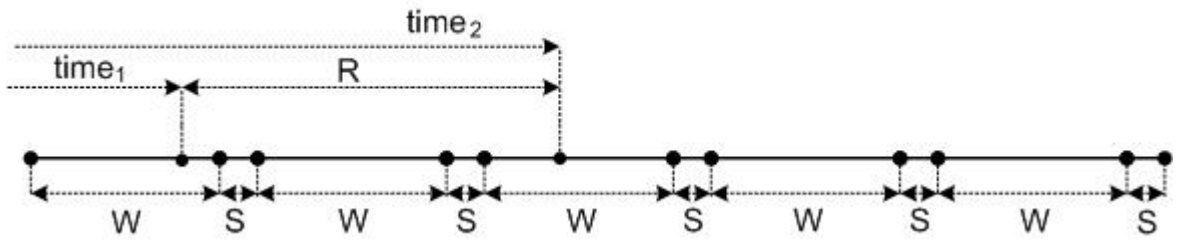

Figure 5.

Now as far as working time $time$ of the station is limited by the value

$$time_{\max} = (N-1)(W+S)+W$$

we should also note that

$$M_{time}(W,S,R) = 1 + \left[\frac{\Re(time) - (t(time)+S)}{W+S}\right] + \delta\left(W,S,\Re(time),t(time)\right)$$

where $\Re(time) = \begin{cases} R & time + R \le time_{\max} \\ time_{\max} - time & else \end{cases}$, see Figure 6.

This means that if the number of cards is limited then if the personalization station crashes while processing the last cards then the number of broken cards can be smaller then usually i.e. even smaller then $M_{\min} = M(W,S,R,W)$.

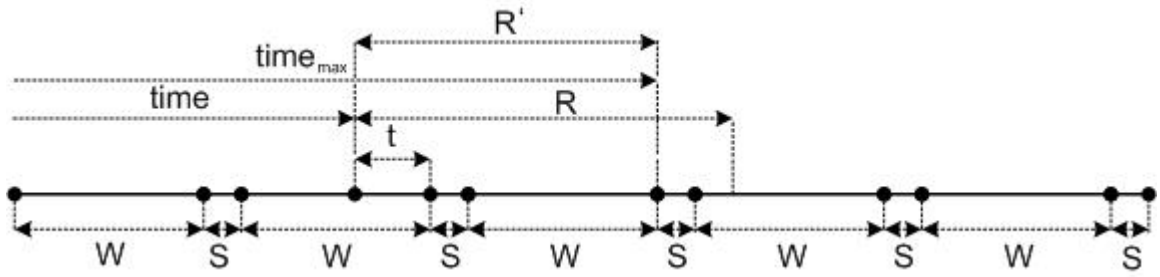

Figure 6.

With the help of formulas defined above we can calculate the set of sets of ordered times $\left\{\left\{time_i^j\right\}_{i=0}^{G_j}\right\}_j$ where $time_0^j = 0$ such that

$$P\left(S_N \le L\right) = \sum_j (1-p)^{G_j}\, p^{\sum_{i=1}^{G_j}\Delta i}$$

and

$$\forall j \quad \sum_{i=1}^{G_j} M_{time_i^j}(W,S,R) \le L$$

Here Δ_i is measured by the following formula:

$$\Delta_i = \begin{cases} \tau + W\left[\dfrac{time_i - \xi\left(time_{i-1}\right) - \tau - S}{W+S}\right], & if\ time_i - \left(time_{i-1}+R\right) - \tau - S \ge 0 \\ \tau, & else \end{cases}$$

where

$$\tau = t\left(\xi\left(time_{i-1}\right)\right) + \left(W - t\left(time_i\right)\right)$$

and

$$\xi(z) = \begin{cases} z+R, & if\ z > 0 \\ 0, & else \end{cases}$$

and presents the amount of working time (station worked) between two successive crashes.

Four personalization stations

The previous result is valid for one personalization station to extend it for 4 stations we should keep in mind that all stations work independently and thus we have the following:

$$P(N\_broken\_cards \le L) = \sum_{\{L_k\}}^{\sum_{k=1}^{4} L_k = L} \prod_{k=1}^{4} P(S_{(N/4)} \le L_k)$$

The last formula assumes that total number of cards in the system equals N and is a multiply of 4. Each station processes the equal number of cards and L_i is the number of cards for i station we expect to be broken.

This formula looks quite ugly and also required a set of sets of ordered times and fragmentations of the expected total number of broken cards L. That is why it can be quite difficult to use.

In order to try to check this formula we can assume that $L = 0$ and thus we have that

$$P(N\_broken\_cards \le 0) = \prod_{k=1}^{4} P(S_{(N/4)} \le L_k) = \prod_{k=1}^{4} p^{W(N/4)} = p^{WN}$$

Of course this is insufficient to prove that it is correct but gives us hope that at least we are on the right way. Also note that this formula works only for the working cycle case when W and S are constant and this is not the case for the model checked in PRISM.

4. Fixed cards model for 4 stations and the schedule

In this section we will present the model with failures in which personalization station can crash each time it takes a new card and then breaks a certain amount of cards before it is repaired.

a. Model description

The main idea of this model is that we involve the fixed number of cards that are broken while the personalization station is broken. The schedule remains to be the SSM.

A personalization station takes card from the belt and we assume that at this moment with probability p it can crash. After crashing personalization station breaks a certain amount of cards and only after this it can crash again.

In this model the number of broken cards doesn't depend from time but depends from the number of cards a personalization station has to process. In this sense as far as stations work independently we can calculate probabilities for one station and then extend them for bigger number of stations.

b. Modeling results

The corresponding model had been created for the PRISM tool and was used with the SSM schedules for 12/30 cards (Appendix C). The resulting plots of gathered probabilities are present below (Figure 7., Figure 8.).

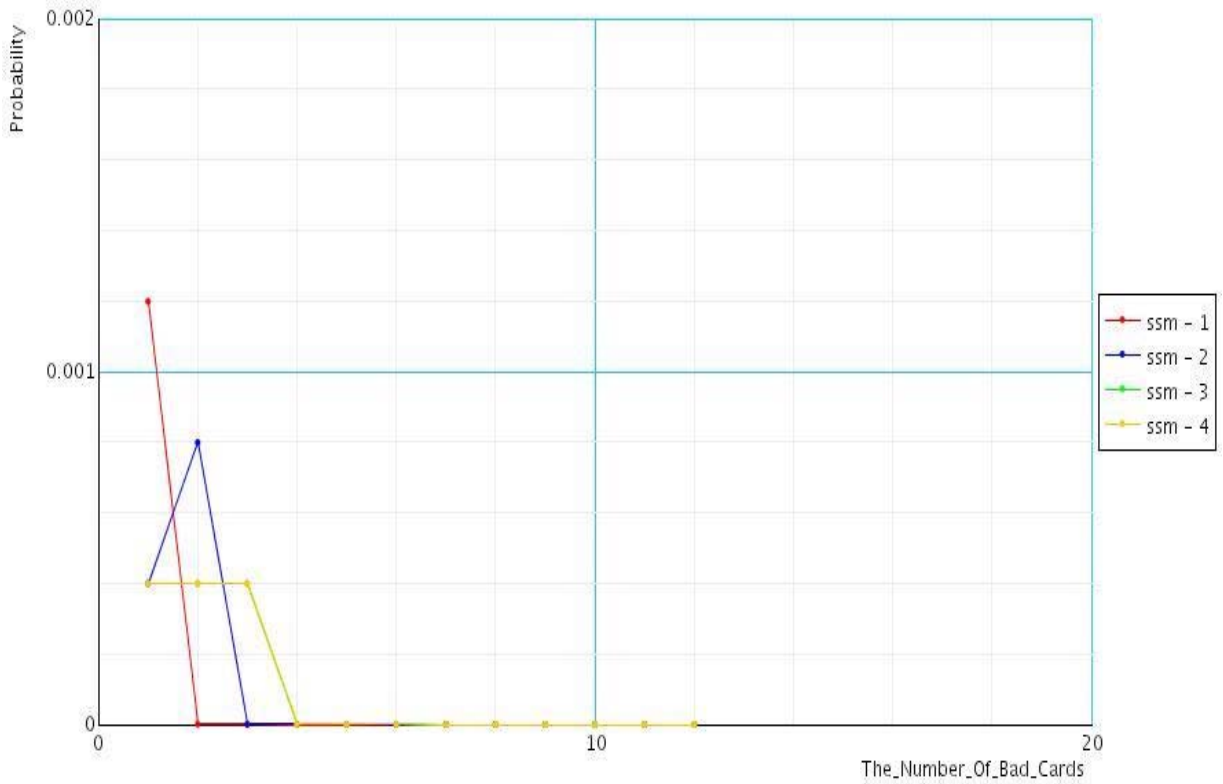

Figure 7. The probability that a certain number of 12 cards will be broken

On the plots above (Figure 7.) we do not represent probability for 0 cards not to obscure the other values. There are four plots on the figure for different number of cards to be broken. For example "ssm–1" means that this is the plot of probabilities for the case when after one crash of personalization station it produces one bad card.

It is interesting to note that plots for "ssm-3"and "ssm-4" are the same. Right now there is no explanation to this.

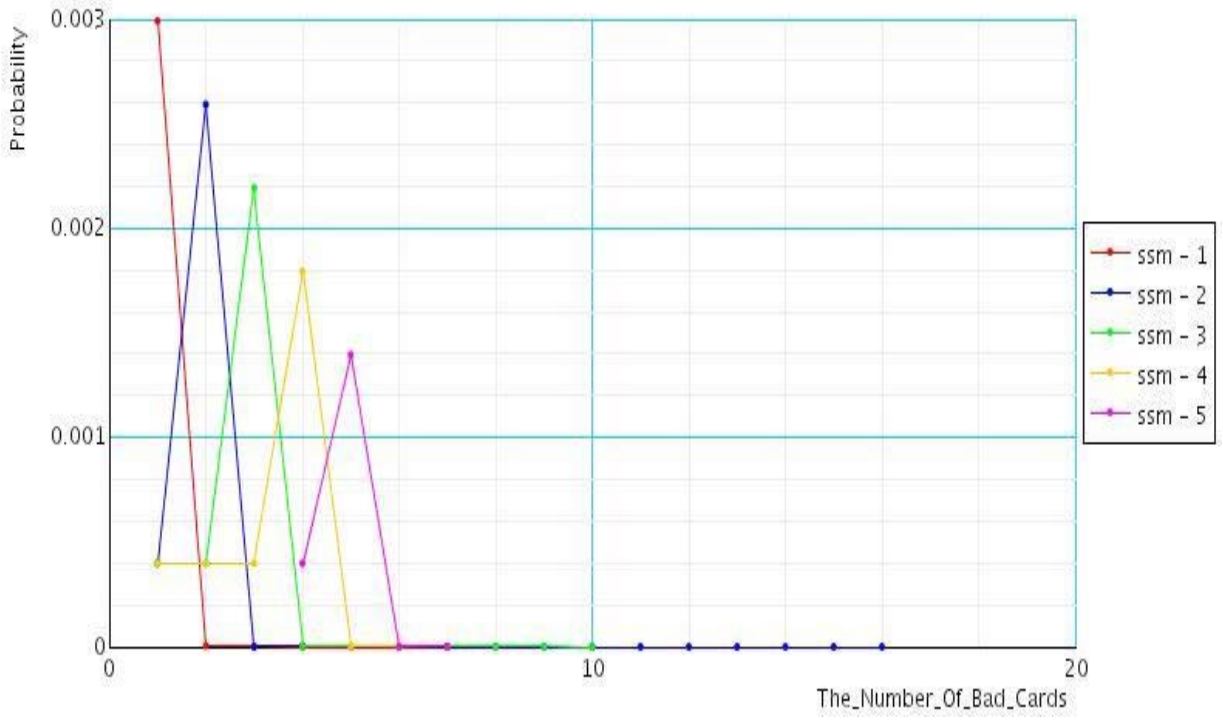

Figure 8. The probability that a certain number of 30 cards will be broken

On the Figure 8 there are also plots for different number of cards to be broken. The moving maximum here is quite obvious as far as the number of cards that are broken after one station crashes affects this probabilities directly.

An open question is why the maximum probabilities are decreasing so rapidly with the increasing of the number of cards that are broken after one crash and why the number of cards before the first maximum is almost constant?

5. Nondeterministic cards model for 4 stations

In this section we will present the model with failures in which personalization station can crash each time it takes a new card and then breaks a nondeterministic amount of cards from the certain interval before it is repaired.

a. Model description

The main idea of this model is that we involve the interval for the number of cards that are broken while the personalization station is broken. Each time a certain number of cards are broken and this amount is selected nondeterministically. The schedule remains to be the SSM.

A personalization station takes card from the belt and we assume that at this moment with probability p it can crash. After crashing personalization station breaks a nondeterministic number of cards within a certain interval after this it can crash again.

This model can also be reduced to calculation of probabilities for one station and then extension this for bigger number of stations. This will be discussed further for "Nondeterministic cards model for one station".

b. Modeling results

The corresponding model had been created for the PRISM tool and was used with the SSM schedules for 12/30 cards (Appendix D). The resulting plots of gathered probabilities are present below (Figure 9.).

Figure 9. The probability that a certain number of 30 cards will be broken

The model is nondeterministic and thus there are maximum and minimum probabilities for each upper bound of cards to be broken.

It is obvious that plots for min probabilities are the same as the lower bound of cards to be broken is always the same.

The question is why for example "ssm max 30/4" is decreasing within interval [1,4] but is not constant?

6. Nondeterministic cards model for one station

Here we reduce model to the one personalization station that can crash and break a nondeterministic number of cards within a certain interval before it is recovered.

a. Model description

There is no belt, no loader and unloader in this model. There is also no schedule and timing as far as station simply takes a card and at this moment decides whether to crash or not if it crashes then it starts to break cards and breaks a nondeterministic amount of cards form a certain interval and then it works properly again.

The results of this model can be extended then to the results of the "Nondeterministic cards model for R stations" as far as stations work independently and the schedule is here is not important as it only provides a station with a certain number of cards to process. Note that with SSM schedule each station processes equal number of cards. So here R is the devisor of the total number of cards.

That is why the results for the model with R stations can be extended via the following formula:

$$P_T^R\left(M \text{ of } N \text{ cards are broken}\right) = \sum_{M_1+\ldots+M_N=M} \prod_{i=1}^{R} P_T^1\left(M_i \text{ of } \frac{N}{4} \text{ cards are broken}\right)$$

Here R is a devisor of N, N is the total number of cards, M is the total number of cards to be broken, $T \in \{\min, \max\}$, P_T^N is the total probability for the R stations, P_T^1 is the calculated before probability for one station.

This formula is also applicable to the deterministic variant where there is the fixed number of cards to be broken; in this case the index T should be removed.

b. Modeling results

The corresponding model had been created for the PRISM tool and was tested with 1000 cards (Appendix E). The resulting plots of gathered probabilities are present below (Figure 10.,Figure 11.).

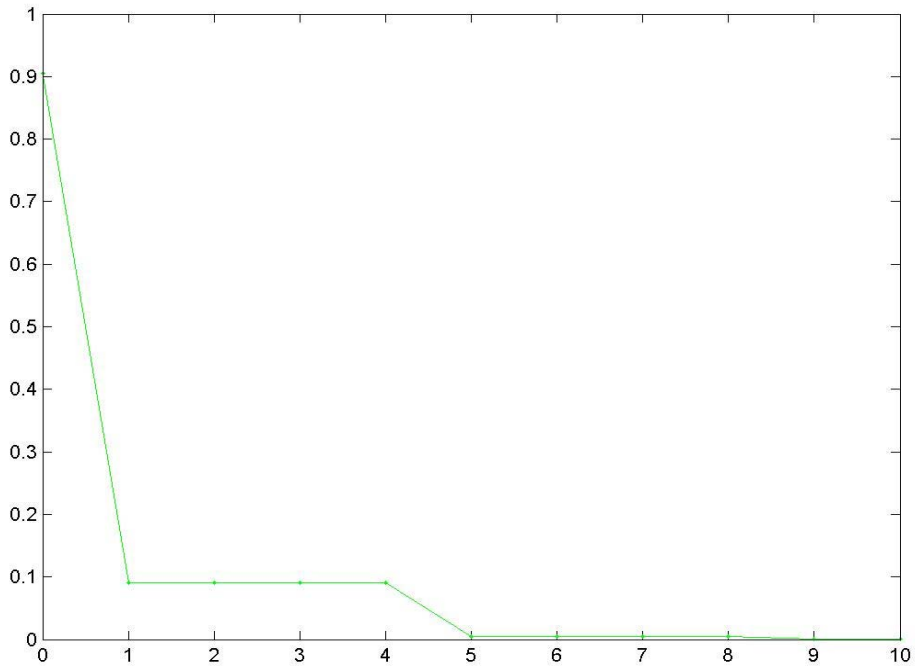

Figure 10. The Max probability that a certain number of 1000 cards will be broken with a maximum 4 cards broken after each crash of personalization station.

Here it is quite obvious that there are stairs with the same probabilities because for 1,2,3 and 4 cards to be broken there is no determinism and the different is made by the slightly different number of combinations of paths that lead to the given number of broken cards. This behavior is periodic.

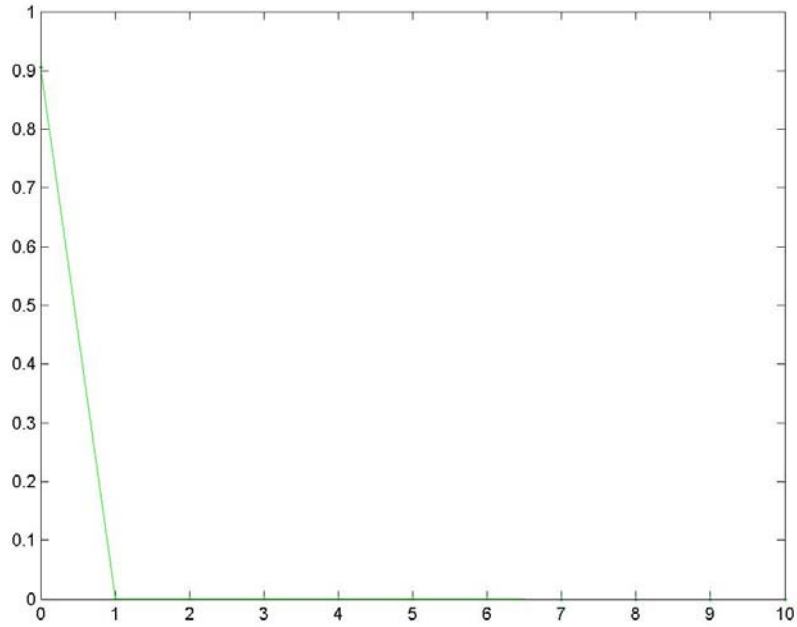

Figure 11. The Min probability that a certain number of 1000 cards will be broken with a maximum 4 cards broken after each crash of personalization station.

Now if we extend the maximum probabilities up to 4 stations then we get the following plot (Figure 12.)

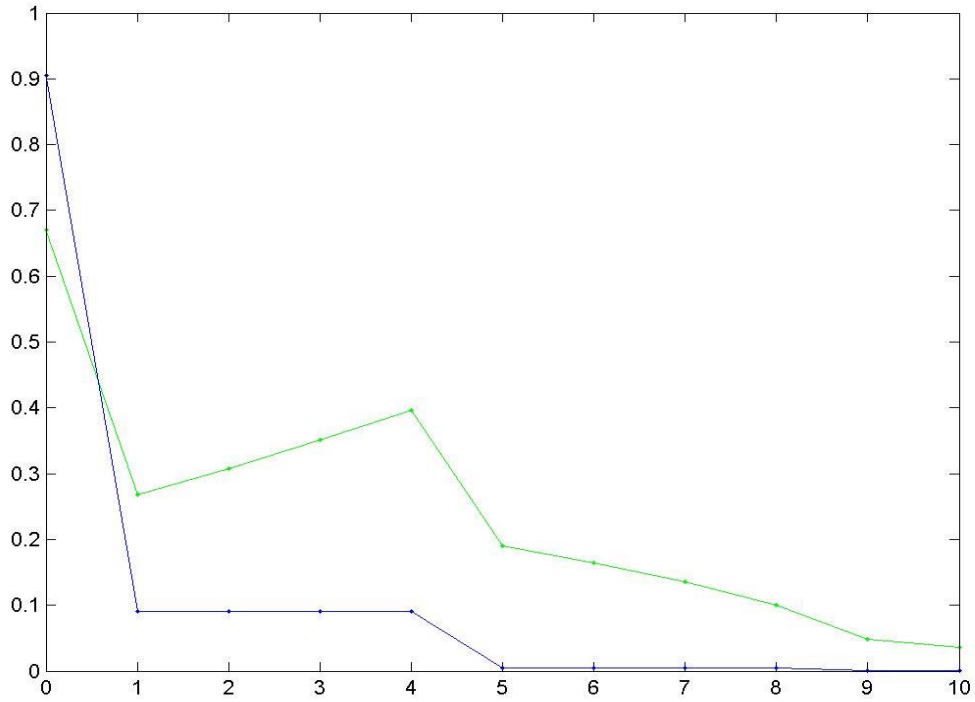

Figure 10. The Max probability that a certain number of 1000 cards will be broken with a maximum 4 cards broken after each crash of personalization station.

On this figure the green plot is the resulting plot and the blue one is the initial plot for one personalization station.

PRISM can't run model with 4 stations, SSM and with 4000 cards that is why the gathered plot can't be verified. We have performed the following experiment to check the extendibility issue.

The idea of the experiment is simple; calculate max probabilities for 7 cards 1 personalization station and the maximum 4 cards broken after crash of personalization station and then extend them up to the 4 personalization stations and 28 cards. After that all we have to do is to compare them with the results gathered for 4 stations and 28 cards. He results are on the figure below (Figure 11.).

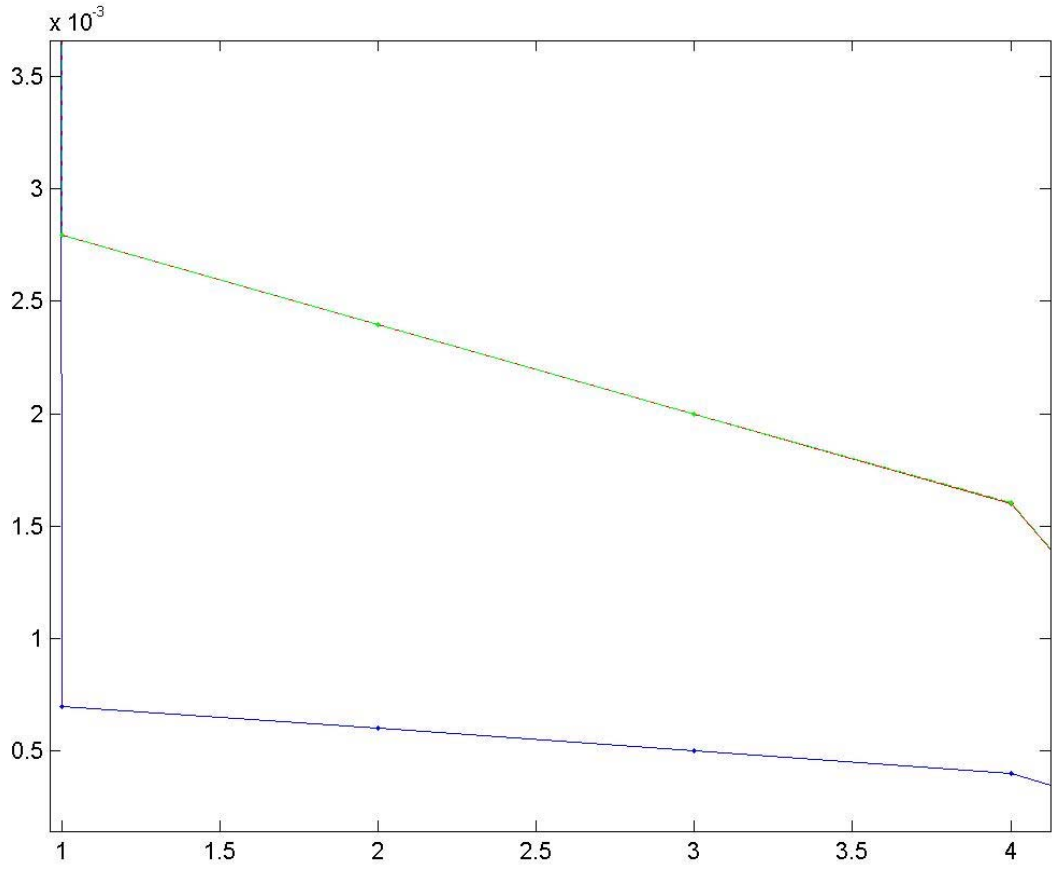

Figure 11. The Max probabilities for 1 station and 7 cards (blue) gathered by PRISM, The Max probabilities for 4 station and 28 cards (green) calculated via formula, The Max probabilities for 4 station and 28 cards (red) gathered by PRISM.

Here we can see that the probabilities calculated by the formula are almost the same as the probabilities gathered by PRIZM. The difference could be explained by the loss of precision.

7. DTMC model for one station

This model is simply a DTMC. We have one personalization station that processes cards one by one and each time it takes a card it can crash with a certain probability, after it crashed it starts to produce broken cards but each time it produces a card it can repair with the certain probability. In other words there is a probability to crash and then a probability to repair.

a. Model description

In this model there are only two states. First state is for the properly working personalization station and a second one for the broken one. The probability to move from the first state to the second one is 0,001. It means that each time station processes a card it can crash and become broken. After this we are in the second state and here there is also a probability to recover that was taken as 0,999.

On the Figure 12 there is a Transition system that represents the model described above.

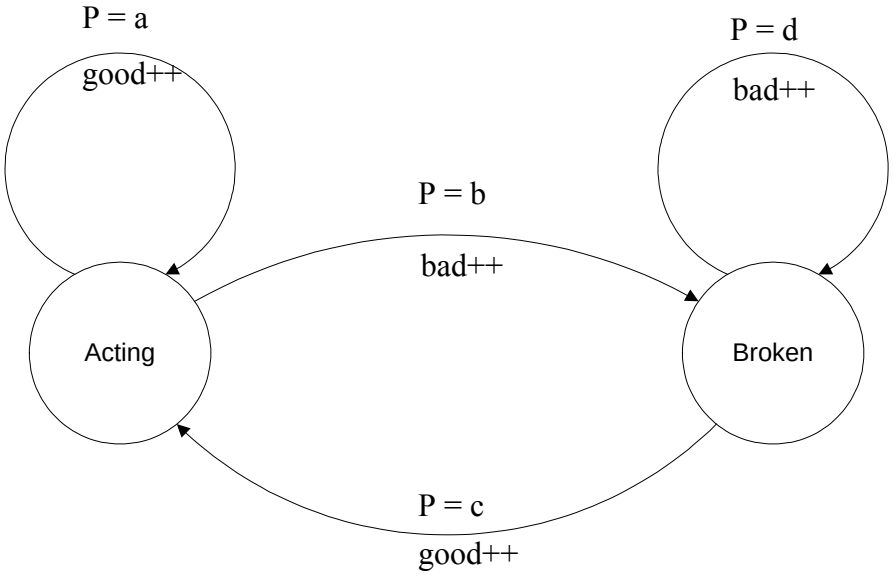

Figure 12. The DTMC for the one personalization station with a probability to recover

The probability values here are present as parameters for generalization.

As far as we have a DTMC we can compute the stable state probabilities. Their calculation is below:

If $\overline{p}_{stable} = \left(p_{good}, p_{bad}\right)$ then we should solve the following set of linear equations:

$$\begin{cases} \overline{p}_{stable}\left(I - \begin{pmatrix} a & b \\ c & d \end{pmatrix}\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ a + b = 1 \\ c + d = 1 \\ p_{good} + p_{bad} = 1 \end{cases}$$

The solution here is simple and is the following:

$$\overline{p}_{stable} = \left(\frac{c}{b+c}, \frac{b}{b+c}\right)$$

In our model we have the following values for the a, b, c, d parameters:

$$\begin{cases} a = 0.999 \\ b = 0.001 \\ c = 0.001 \\ d = 0.999 \end{cases}$$

Knowing the stable state probabilities we can forecast the number of broken cards after the work of this personalization on the certain amount of cards. Lets say we have 1000 a cards then:

$$\overline{p}_{stable} = \left(p_{good}, p_{bad}\right) = \left(\frac{c}{b+c}, \frac{b}{b+c}\right) = \left(0.999, 0.001\right)$$

That is why after processing 1000 cards we can presume that there will be one broken card.

b. Modeling results

The corresponding model had been created for the PRISM tool and was tested with 1000 cards (Appendix F). The resulting plots of gathered probabilities are present below (Figure 13.).

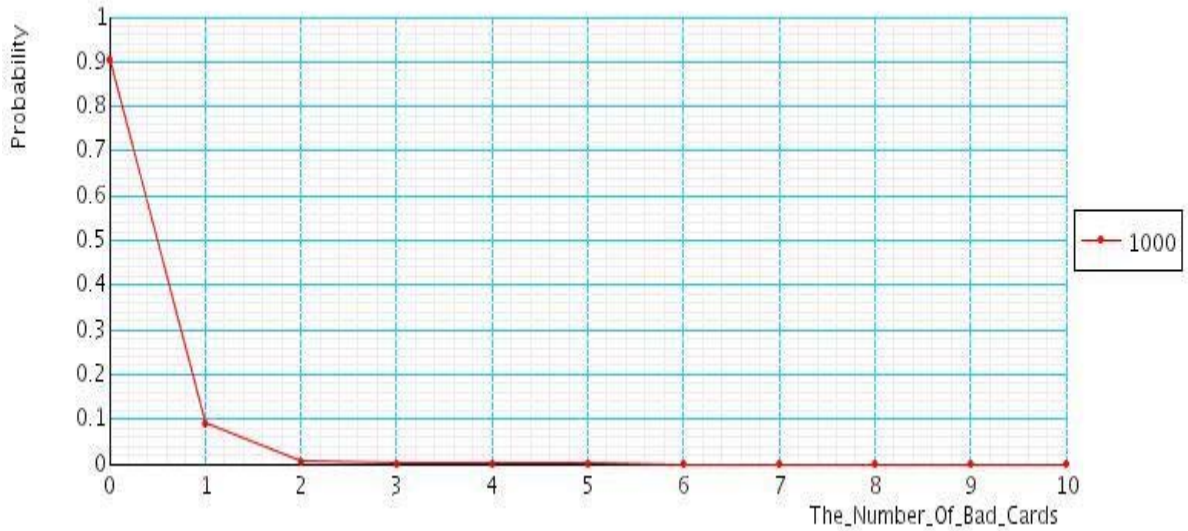

Figure13. The probability that a certain number of 1000 cards will be broken

8. Fixed cards model for one station

In this section we will present the model with failures in which there is one personalization station and it can crash each time it takes a new card and then breaks a certain amount of cards before it is repaired.

a. Model description

The main idea of this model is that we involve the fixed number of cards that are broken while the personalization station is broken. There is no schedule and there is only one personalization station.

A personalization station takes card and we assume that at this moment with probability p it can crash. After crashing personalization station breaks a certain amount of cards and only after this it can crash again.

In this model there is no timing and scheduling and that is why it is represented by a simple automata. In this sense if we assume that we have several stations and they work independently we can calculate probabilities for one station and then extend them to a bigger number of stations.

b. Modeling results

The corresponding model had been created for the PRISM tool and was tested with 1000 cards (Appendix G). The resulting plots of gathered probabilities are present below (Figure 14.).

On this figure there are 4 different plots each of them for a certain fixed number of cards to be broken after personalization station crashes. For instance “1000/2” means that personalization station was tested with 1000 cards and the fixed number of cards to be broken after one crash was equal 2.

We can see a periodical behavior of plots i.e. there are local maximums in the points that are divisible by the number of cards to be broken and this is an expected behavior.

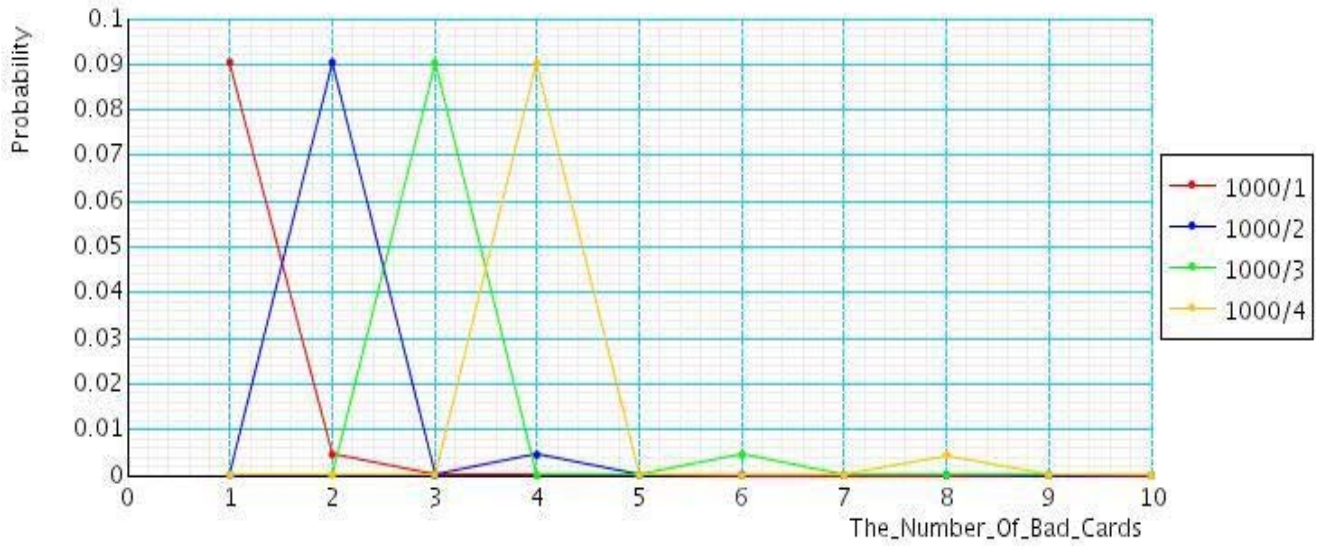

Figure 14. The probability that a certain number of 1000 cards will be broken

9. Weibull based model for one station

In this section we will present the model with failures in which there is one personalization station and it breaks cards depending on the probability defined by the variant of the ***Weibull discrete distribution***.

a. Model description

This model is almost the same as the initial Simple Failure model but there is one significant difference. The probability to break a card depends on the number of good cards k produced since the last failure. The distribution that is used is called the ***type I discrete Weibull distribution*** and is defined by the following reliability function $R(k)$ and the failure rate $\lambda(k)$:

$$R(k) = q^{k^\beta},$$

$$\lambda(k) = 1 - q^{k^\beta - (k-1)^\beta}$$

Where $q \in]0,1[$, $\beta > 0, k \in N^*$. If $\beta > 1$ then the distribution is IFR (Increasing Failure Rate distribution i.e. the failure rate increases with increasing of k).

As far as we need to implement the wear out of the personalization station we need exactly that type of distribution.

To make things clear let us explain the probabilistic meaning of the reliability function $R(k)$ and failure rate $\lambda(k)$:

Let $p(k) = P(K = k)$ be the probability of failure of the device on any demand k (K is a random variable – the number of demands until the first failure). Then the reliability function $R(k) = P(K > k)$ gives the probability that the device is still alive at demand k. The failure rate $\lambda(k)$ is then defined as $\lambda(k) = P(K = k \mid K \geq k) = \frac{p(k)}{R(k-1)}$ and gives the conditional probability of failure of the device on the demand k given that it didn't fail before.

We should note that after each failure of personalization station the wear out process starts from the very beginning i.e. like the personalization station is immediately replaced by a new one. The probability of failure at each k is defined by the $\lambda(k)$ (k is the number of good cards produced since the last failure) and $\lambda(1) = 1 - q$ where q is the probability not to fail at the first demand (while personalizing of the first card).

The only one problem here is to select the value of β. Originally there are several statistical methods to estimate the parameter value which are based on the statistical analysis but as far as we have no information about the wear out of personalization station except that it breaks at least 1 card among 1000 cards we can take $\beta = 2.2, q = 0.999$.

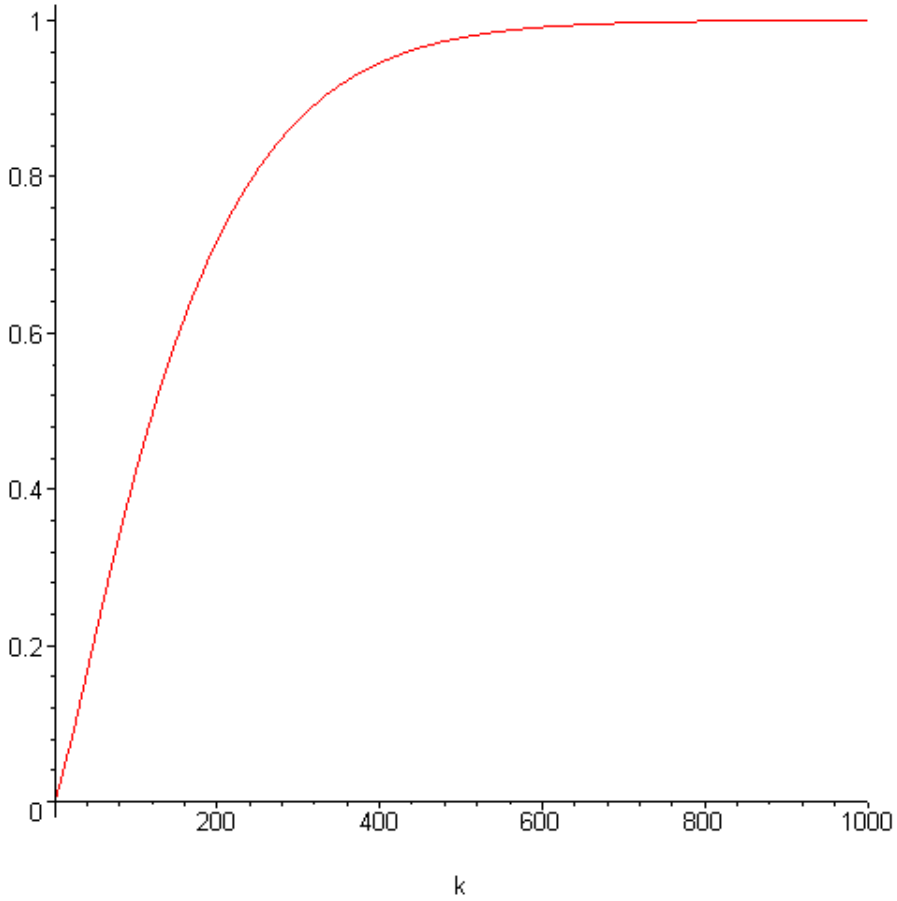

Figure 15. The $\lambda(k)$ for $\beta = 2.2, q = 0.999$

On the Figure 15 there is a plot that shows behavior of $\lambda(k)$ for $\beta = 2.2, q = 0.999$. The idea is that the probability to fail after producing 999 good cards is almost 1.

b. Modeling results

The corresponding model had been created for the PRISM tool and was run with 500,200,100 and 50 cards (Appendix H). The resulting plots of gathered probabilities are presented below (Figure 16.).

On this figure there are 3 different plots each of them for a different number of input cards (50/100/200).

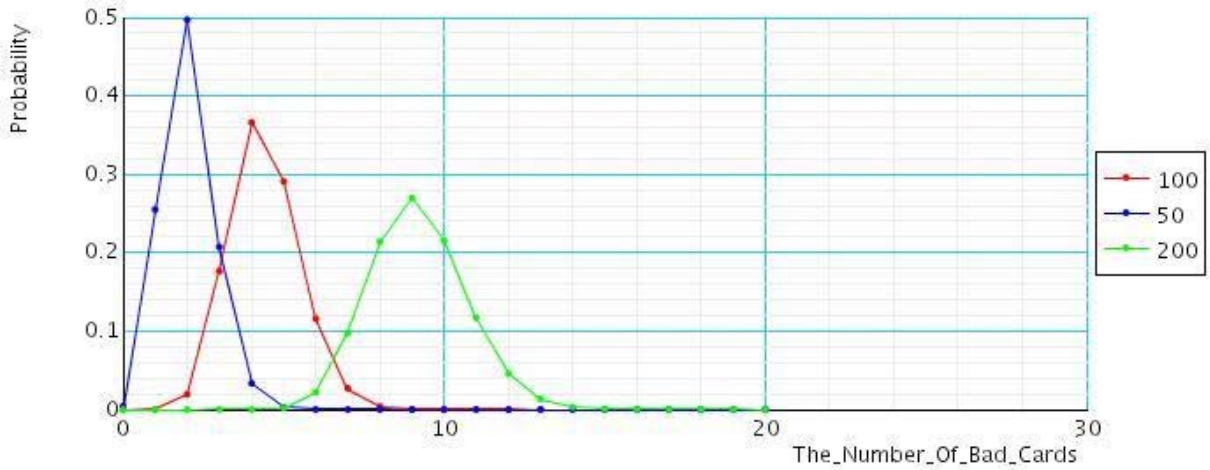

Figure 16. The probability that a certain number of 200/100/50 cards will be broken

10. Conclusions

In this report we described the different models that emulate the personalization stations failures. There are some experimental results gathered for these models and also theoretical derivations the aim of which was to investigate the models in depth and to create formulas for calculation of required probabilities. In the simplest cases (Simple Failure Model, DTMC) we were able to create the formulas that give analytical results, which conform the results gathered by PRISM. For the more complicated cases this is still a matter of further investigation.

11. Future work

In the future the following issues could be studied:

- There are still issues to be explained in the gathered results,
- Some of presented models can be tried to be solved analytically,

12. Appendix A.

```
nondeterministic
//The number of cards
const int N = 12;
//The empty cell value
const int EMPTY_CELL = 0;
//The cell with a card
const int WITH_CARD_CELL = 1;
//The cell with a bad card
const int BROKEN_CARD_CELL = 2;

module unload
  unloadCell : [EMPTY_CELL..WITH_CARD_CELL];

  [unload] true -> (unloadCell'=WITH_CARD_CELL);

  [tick] !belt_is_going_to_move -> (unloadCell'=unloadCell);
  [tick] belt_is_going_to_move -> (unloadCell'=EMPTY_CELL);
endmodule

//True if we try to unload the bad card
formula bad_card =  loadCell = BROKEN_CARD_CELL;

module load
  loadCell : [EMPTY_CELL..BROKEN_CARD_CELL];
```

```
  bad_cards_counter : [0..N];

  [load] !bad_card -> (loadCell'=EMPTY_CELL);
  [load] bad_card -> (loadCell'=EMPTY_CELL) & (bad_cards_counter'=bad_cards_counter+1);

  [tick] !belt_is_going_to_move -> (loadCell'=loadCell);
  [tick] belt_is_going_to_move -> (loadCell'=v4Cell);
endmodule

module v1
  v1Cell : [EMPTY_CELL..BROKEN_CARD_CELL];

  //Get the card from the belt (it takes no time)
  [get_v1] true -> (v1Cell'=EMPTY_CELL);
  //Put the card to the belt
  [put_v1] true -> 0.999 : (v1Cell'=WITH_CARD_CELL) + 0.001 : (v1Cell'=BROKEN_CARD_CELL);

  [tick] !belt_is_going_to_move -> (v1Cell'=v1Cell);
  [tick] belt_is_going_to_move -> (v1Cell'=unloadCell);
endmodule

module v2 = v1 [v1Cell = v2Cell, unloadCell = v1Cell ] endmodule

module v3 = v1 [v1Cell = v3Cell, unloadCell = v2Cell ] endmodule

module v4 = v1 [ v1Cell = v4Cell, unloadCell = v3Cell ] endmodule

module belt
  belt_is_going_to_move : bool init false;

  //Decide that the belt if going to move
  [shift] true -> (belt_is_going_to_move'=true);

  //Reset the flag value. It doesn't matter whether the belt is going to move or not.
  [tick] true -> (belt_is_going_to_move'=false);
endmodule

module scheduler
  //12 cards, 4 stations work.
endmodule

system
  scheduler || belt || unload || load || v1 || v2 {get_v1 <- get_v2, put_v1 <- put_v2} || v3 {get_v1 <-
get_v3, put_v1 <- put_v3} || v4 {get_v1 <- get_v4, put_v1 <- put_v4}
endsystem
```

13. Appendix B.

```
nondeterministic
//The number of cards
const int N = 12;
//The empty cell value
const int EMPTY_CELL = 0;
//The cell with a card
const int WITH_CARD_CELL = 1;
//The cell with a bad card
const int BROKEN_CARD_CELL = 2;
//The time to fix the station;
const int FIX_TIME = 25;

module unload
  unloadCell : [EMPTY_CELL..WITH_CARD_CELL];
```

```
  [unload] true -> (unloadCell'=WITH_CARD_CELL);

  [tick] !belt_is_going_to_move -> (unloadCell'=unloadCell);
  [tick] belt_is_going_to_move -> (unloadCell'=EMPTY_CELL);
endmodule

//True if we try to unload the bad card
formula bad_card =  loadCell = BROKEN_CARD_CELL;

module load
  loadCell : [EMPTY_CELL..BROKEN_CARD_CELL];

  bad_cards_counter : [0..N];

  [load] !bad_card -> (loadCell'=EMPTY_CELL);
  [load] bad_card -> (loadCell'=EMPTY_CELL) & (bad_cards_counter'=bad_cards_counter+1);

  [tick] !belt_is_going_to_move -> (loadCell'=loadCell);
  [tick] belt_is_going_to_move -> (loadCell'=v4Cell);
endmodule

module v1
  v1Cell : [EMPTY_CELL..BROKEN_CARD_CELL];
  v1_is_working : bool init false;
  v1_counter : [0..FIX_TIME];

  //Get the card from the belt (it takes no time)
  [get_v1] true -> (v1Cell'=EMPTY_CELL) & (v1_is_working'=true);
  //Put the card to the belt
  [put_v1] v1_counter=FIX_TIME -> (v1Cell'=BROKEN_CARD_CELL) & (v1_is_working'=false) & (v1_counter'=0);
  [put_v1] v1_counter>0 & v1_counter<FIX_TIME -> (v1Cell'=BROKEN_CARD_CELL) & (v1_is_working'=false);
  [put_v1] v1_counter=0 -> (v1Cell'=WITH_CARD_CELL) & (v1_is_working'=false);

  //If we are not working then we can not break
  //If we are not working but have been fixed already then indicate this.
  [tick] ( v1_counter=0 | v1_counter=FIX_TIME ) & !v1_is_working & !belt_is_going_to_move -> (v1Cell'=v1Cell) & (v1_counter'=0);
  [tick] ( v1_counter=0 | v1_counter=FIX_TIME ) & !v1_is_working & belt_is_going_to_move -> (v1Cell'=unloadCell) & (v1_counter'=0);

  //If we are working and have been fixed already then wait until put to indicate this
  [tick] (v1_counter=0 | v1_counter=FIX_TIME) & v1_is_working & !belt_is_going_to_move -> 0.9999: (v1Cell'=v1Cell) + 0.0001: (v1Cell'=v1Cell) & (v1_counter'=1);
  [tick] (v1_counter=0 | v1_counter=FIX_TIME) & v1_is_working & belt_is_going_to_move -> 0.9999: (v1Cell'=unloadCell) + 0.0001: (v1Cell'=unloadCell) & (v1_counter'=1);

  //If we have been breaken then we are fixing in any case
  [tick] v1_counter>0 & v1_counter<FIX_TIME & !belt_is_going_to_move -> (v1Cell'=v1Cell) & (v1_counter'=v1_counter+1);
  [tick] v1_counter>0 & v1_counter<FIX_TIME & belt_is_going_to_move -> (v1Cell'=unloadCell) & (v1_counter'=v1_counter+1);

endmodule

module v2 = v1 [v1_is_working=v2_is_working, v1_counter=v2_counter, v1Cell = v2Cell, unloadCell = v1Cell ] endmodule

module v3 = v1 [v1_is_working=v3_is_working, v1_counter=v3_counter, v1Cell = v3Cell, unloadCell = v2Cell ] endmodule
```

```
module v4 = v1 [v1_is_working=v4_is_working, v1_counter=v4_counter, v1Cell = v4Cell, unloadCell = v3Cell ] endmodule

module belt
  belt_is_going_to_move : bool init false;

  //Decide that the belt if going to move
  [shift] true -> (belt_is_going_to_move'=true);

  //Reset the flag value. It doesn't matter whether the belt is going to move or not.
  [tick] true -> (belt_is_going_to_move'=false);
endmodule

module scheduler
  //12 cards, 4 stations work.
endmodule

system
  scheduler || belt || unload || load || v1 || v2 {get_v1 <- get_v2, put_v1 <- put_v2} || v3 {get_v1 <- get_v3, put_v1 <- put_v3} || v4 {get_v1 <- get_v4, put_v1 <- put_v4}
endsystem
```

14. Appendix C.

```
nondeterministic
//The number of cards
const int N = 12;
//The empty cell value
const int EMPTY_CELL = 0;
//The cell with a card
const int WITH_CARD_CELL = 1;
//The cell with a bad card
const int BROKEN_CARD_CELL = 2;
//The the number of cards to be BROKEN after single personalization station crash
const int NUMBER_OF_CARDS_TO_BE_BROKEN;

module unload
  unloadCell : [EMPTY_CELL..WITH_CARD_CELL];

  [unload] true -> (unloadCell'=WITH_CARD_CELL);

  [tick] !belt_is_going_to_move -> (unloadCell'=unloadCell);
  [tick] belt_is_going_to_move -> (unloadCell'=EMPTY_CELL);
endmodule

//True if we try to unload the bad card
formula bad_card =  loadCell = BROKEN_CARD_CELL;

module load
  loadCell : [EMPTY_CELL..BROKEN_CARD_CELL];

  bad_cards_counter : [0..N];

  [load] !bad_card -> (loadCell'=EMPTY_CELL);
  [load] bad_card -> (loadCell'=EMPTY_CELL) & (bad_cards_counter'=bad_cards_counter+1);

  [tick] !belt_is_going_to_move -> (loadCell'=loadCell);
  [tick] belt_is_going_to_move -> (loadCell'=v4Cell);
endmodule

module v1
  v1Cell : [EMPTY_CELL..BROKEN_CARD_CELL];
```

```
  v1_counter : [0..NUMBER_OF_CARDS_TO_BE_BROKEN];

  //Get the card from the belt (it takes no time)
  [get_v1] true -> 0.9999: (v1Cell'=EMPTY_CELL) + 0.0001: (v1Cell'=EMPTY_CELL) &
(v1_counter'=1);

  //Put the card to the belt
  [put_v1] v1_counter=0 -> (v1Cell'=WITH_CARD_CELL);
  [put_v1] v1_counter>0 & v1_counter<NUMBER_OF_CARDS_TO_BE_BROKEN ->
(v1Cell'=BROKEN_CARD_CELL) & (v1_counter'=v1_counter+1);
  [put_v1] v1_counter=NUMBER_OF_CARDS_TO_BE_BROKEN ->
(v1Cell'=BROKEN_CARD_CELL) & (v1_counter'=0);

  [tick] !belt_is_going_to_move -> (v1Cell'=v1Cell);
  [tick] belt_is_going_to_move -> (v1Cell'=unloadCell);

endmodule

module v2 = v1 [v1_counter=v2_counter, v1Cell = v2Cell, unloadCell = v1Cell ] endmodule

module v3 = v1 [v1_counter=v3_counter, v1Cell = v3Cell, unloadCell = v2Cell ] endmodule

module v4 = v1 [v1_counter=v4_counter, v1Cell = v4Cell, unloadCell = v3Cell ] endmodule

module belt
  belt_is_going_to_move : bool init false;

  //Decide that the belt if going to move
  [shift] true -> (belt_is_going_to_move'=true);

  //Reset the flag value. It doesn't matter whether the belt is going to move or not.
  [tick] true -> (belt_is_going_to_move'=false);
endmodule

module scheduler
  //Here should be a ssm schedule for 12 cards

endmodule

system
  scheduler || belt || unload || load || v1 || v2 {get_v1 <- get_v2, put_v1 <- put_v2} || v3 {get_v1 <-
get_v3, put_v1 <- put_v3} || v4 {get_v1 <- get_v4, put_v1 <- put_v4}
endsystem
```

15. Appendix D.

```
nondeterministic
//The number of cards
const int N = 12;
//The empty cell value
const int EMPTY_CELL = 0;
//The cell with a card
const int WITH_CARD_CELL = 1;
//The cell with a bad card
const int BROKEN_CARD_CELL = 2;
//The max number of cards to be broken after single personalization station crash
const int MAX_NUMBER_OF_CARDS_TO_BE_BROKEN;

module unload
  unloadCell : [EMPTY_CELL..WITH_CARD_CELL];

  [unload] true -> (unloadCell'=WITH_CARD_CELL);
```

```
  [tick] !belt_is_going_to_move -> (unloadCell'=unloadCell);
  [tick] belt_is_going_to_move -> (unloadCell'=EMPTY_CELL);
endmodule

//True if we try to unload the bad card
formula bad_card =  loadCell = BROKEN_CARD_CELL;

module load
  loadCell : [EMPTY_CELL..BROKEN_CARD_CELL];

  bad_cards_counter : [0..N];

  [load] !bad_card -> (loadCell'=EMPTY_CELL);
  [load] bad_card -> (loadCell'=EMPTY_CELL) & (bad_cards_counter'=bad_cards_counter+1);

  [tick] !belt_is_going_to_move -> (loadCell'=loadCell);
  [tick] belt_is_going_to_move -> (loadCell'=v4Cell);
endmodule

module v1
  v1Cell : [EMPTY_CELL..BROKEN_CARD_CELL];
  v1_counter : [0..MAX_NUMBER_OF_CARDS_TO_BE_BROKEN];

  //Get the card from the belt (it takes no time)
  [get_v1] true -> 0.9999: (v1Cell'=EMPTY_CELL) + 0.0001: (v1Cell'=EMPTY_CELL) &
(v1_counter'=1);

  //Put the card to the belt
  [put_v1] v1_counter=0 -> (v1Cell'=WITH_CARD_CELL);

  [put_v1] v1_counter>0 & v1_counter<MAX_NUMBER_OF_CARDS_TO_BE_BROKEN ->
(v1Cell'=BROKEN_CARD_CELL) & (v1_counter'=v1_counter+1);
  [put_v1] v1_counter>0 & v1_counter<MAX_NUMBER_OF_CARDS_TO_BE_BROKEN ->
(v1Cell'=BROKEN_CARD_CELL) & (v1_counter'=0);

  [put_v1] v1_counter=MAX_NUMBER_OF_CARDS_TO_BE_BROKEN ->
(v1Cell'=BROKEN_CARD_CELL) & (v1_counter'=0);

  [tick] !belt_is_going_to_move -> (v1Cell'=v1Cell);
  [tick] belt_is_going_to_move -> (v1Cell'=unloadCell);

endmodule

module v2 = v1 [v1_counter=v2_counter, v1Cell = v2Cell, unloadCell = v1Cell ] endmodule

module v3 = v1 [v1_counter=v3_counter, v1Cell = v3Cell, unloadCell = v2Cell ] endmodule

module v4 = v1 [v1_counter=v4_counter, v1Cell = v4Cell, unloadCell = v3Cell ] endmodule

module belt
  belt_is_going_to_move : bool init false;

  //Decide that the belt if going to move
  [shift] true -> (belt_is_going_to_move'=true);

  //Reset the flag value. It doesn't matter whether the belt is going to move or not.
  [tick] true -> (belt_is_going_to_move'=false);
endmodule

module scheduler
  //Here should be a ssm schedule for 12 cards
```

```
endmodule

system
  scheduler || belt || unload || load || v1 || v2 {get_v1 <- get_v2, put_v1 <- put_v2} || v3 {get_v1 <-
get_v3, put_v1 <- put_v3} || v4 {get_v1 <- get_v4, put_v1 <- put_v4}
endsystem
```

16. Appendix E.

```
nondeterministic
//The number of cards
const int N;
const int MAX_NUM_OF_BROKEN_CARDS;

module v1
  good : [0..N];
  bad : [0..N];
  state : [0..MAX_NUM_OF_BROKEN_CARDS];

  [] (state=0) & (good+bad<N) -> 0.9999: (good'=good+1) + 0.0001: (state'=state+1) & (bad'=bad+1);

  [] (state>0) & (state<MAX_NUM_OF_BROKEN_CARDS) & (good+bad<N) -> (state'=state+1) &
(bad'=bad+1);
  [] (state>0) & (state<=MAX_NUM_OF_BROKEN_CARDS) & (good+bad<N) -> (state'=0);

  [] (good+bad=N) -> (good'=good);

endmodule
```

17. Appendix F.

```
stochastic
//The number of cards
const int N;
//The Probability for station to crash
const double P_CRASH = 0.9999;
//The Probability for station not to crash
const double P_N_CRASH = 1-P_CRASH;
//The Probability for station to recover
const double P_RECOVER = 0.9999;
//The Probability for station not to recover
const double P_N_RECOVER = 1-P_RECOVER;

//States markers
const int WORK = 0;
const int FAIL = 1;

module v1
  good : [0..N];
  bad : [0..N];
  state : [0..1];

  //If the station works correctly then it can crash
  [] (state=WORK) & (good+bad<N) -> P_CRASH: (good'=good+1) + P_N_CRASH: (state'=FAIL) &
(bad'=bad+1);

  //If the station is crashed then it can be repaired
  [] (state=FAIL) & (good+bad<N) -> P_RECOVER: (state'=WORK) & (good'=good+1) + P_N_RECOVER:
(bad'=bad+1);

  [] (good+bad=N) -> (good'=good);
```

```
endmodule
```

18. Appendix G.

```
stochastic
//The number of cards
const int N;
//The Probability for station to crash
const double P_CRASH = 0.9999;
//The Probability for station not to crash
const double P_N_CRASH = 1-P_CRASH;

const int MAX_NUMBER_OF_BROKEN_CARDS;

module v1
  good : [0..N];
  bad : [0..N];
  state : [0..MAX_NUMBER_OF_BROKEN_CARDS];

  //If the station works then it can crash
  [] (state=0) & (good+bad<N) -> P_CRASH: (good'=good+1) + P_N_CRASH: (state'=1) & (bad'=bad+1);

  //If the station was crashed break cards
  [] (state>0) & (state<MAX_NUMBER_OF_BROKEN_CARDS) & (good+bad<N) -> (bad'=bad+1) &
(state'=state+1);

  //Return to initial state
  [] (state=MAX_NUMBER_OF_BROKEN_CARDS) & (good+bad<N) -> (state'=0);

  [] (good+bad=N) -> (good'=good);

endmodule
```

19. Appendix H.

```
    //The number of cards
    const int N;

    module v1
      num_good : [0..N];
      good : [0..N];
      bad : [0..N];

    //If the station works correctly then it can crash
    [] (good+bad<N) & (num_good=0) -> 0.999: (num_good'=num_good+1) & (good'=good+1) + 0.001:
    (num_good'=0) & (bad'=bad+1);

    …..

    [] (good+bad<N) & (num_good=499) -> 0.02215642796437467: (num_good'=num_good+1) &
    (good'=good+1) + 0.9778435720356253: (num_good'=0) & (bad'=bad+1);

      [] (good+bad=N) -> (good'=good);

    endmodule
```