Dependability and Survivability Evaluation of a Water Distribution Process with Arcade

Stephan Roolvink
DACS, University of Twente
The Netherlands
s.roolvink@ewi.utwente.nl

Anne Remke
DACS, University of Twente
The Netherlands
anne@cs.utwente.nl

Mariëlle Stoelinga
FMT, University of Twente
The Netherlands
marielle@cs.utwente.nl

***Abstract*—Among others, drinking water belongs to the so-called critical infrastructures. To ensure that the water production meets current and future societal needs, a systematic and rigorous analysis is needed. In this paper, we report our first experience with dependability analysis of the last phase of a water treatment facility, namely the water distribution. We use the architectural language Arcade to model this facility and use the Arcade toolset to compute three relevant dependability measures: the *availability* of the water distribution, the *reliability*, i.e., the probability that the water distribution fails, and the *survivability*, that is, the ability to recover from disasters. Since survivability is not directly expressible in the Arcade formalism, we show how one can modify the toolchain for the analysis of survivability.**

I. Introduction

In a recent report of the Dutch Ministry for Internal Affairs [6], water cleaning and distribution has been identified as one of thirteen critical infrastructures. They all include assets that are essential for the functioning of a society and economy. Hence, it is very important that critical infrastructures survive catastrophic events. A water treatment facility cleans raw water in approximately fifteen steps before the water is distributed to households and companies. In case the facility fails to provide clean water it will be charged high fines by the companies it is supposed to deliver to, and it will suffer damage to its public image.

This paper focuses on the distribution station of a water treatment facility, for which *availability*, the readiness for correct service [1], *reliability*, the continuity of correct service [1] and *survivability*, the ability to recover [4], are analyzed. Using the Arcade tool [2] an Input/Output interactive Markov chain (I/O-IMC) model of the distribution station is defined. While availability and reliability can be readily computed within Arcade, we show in the following how the toolset can be enhanced to compute survivability measures. In [2] the availability and reliability of a distributed database system and of a reactor cooling system have been computed using Arcade. In [4] the survivability of the Google file-system is computed using a continuous stochastic logic (CSL) model checking approach. Until now, however, Arcade has never been used to compute survivability type of measures.

Organization of the paper. This paper is organized as follows. In Section II we introduce the water distribution process and in Section III Arcade is shortly described. Section V describes the measures of interest, and how they can be computed within Arcade. In Section VI we discuss the analytical results before Section VII concludes this paper.

II. Water distribution process

Figure 1 depicts the global structure of the water distribution process. It consists of the two water reservoirs, a pumping station, and a distribution station. The reservoirs are used to store the cleaned drinking water before it is pumped, by the pumping station, to the distribution station, which distributes the water to two different water districts.

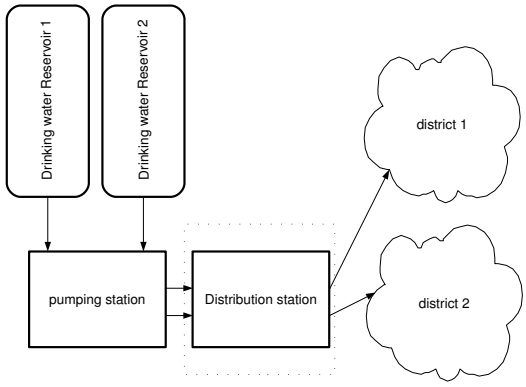


Fig. 1. Storage and water distribution within the water treatment facility.

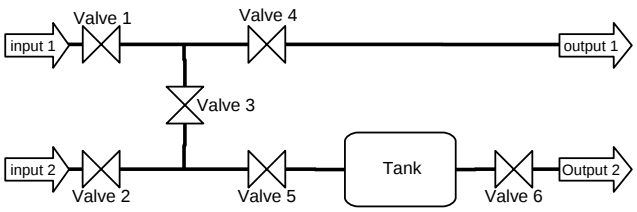


Fig. 2. Water distribution.

III. Arcade

Arcade is an architectural language developed by Crouzen et al.[2] to model and analyze dependability at a system level.

Arcade modeling. Arcade models a system using three basic buildingblocks: (i) a Basic Component (BC), (ii) a Repair Unit (RU) and (iii) a Spare Management Unit (SMU).

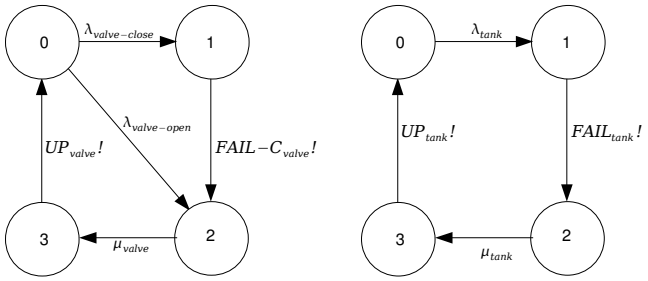
A BC is a model of a physical or logical system component. For each system component that is modeled, a BC is defined, describing the corresponding operation and failure modes of the component. For example, a pump can fail by not working, a valve, on the other hand, can fail by being stuck in open or closed position. The RU defines the way in which the repair of a BC is organized. Different BCs can have the same RU. This happens, for example, when faults occur at different BCs that belong to the same RU. The RU will then schedule the order in which the repairs are performed. Components can have spares that can take over their functionality. The SMU handles the usage of these spare components. When the primary component fails, a spare component is activated. When the primary has been repaired, the SMU deactivates the active spare component. Arcade describes the system failures by means of a fault tree (FT).

Arcade analysis. Arcade translates the buildingblocks to Input/Output Interactive Markov Chains (I/O-IMCs) for their analysis, and applies the powerful compositional aggregation technique to generate the underlying Continuous Time Markov Chain (CTMC). In some cases, non-determinism may be present in the system, so that a continuous-time Markov decision process is generated. Non-determinism is easy to detect, and did not occur in our models. That is, Arcade composes the I/O-IMC buildingblocks one-by-one and applies minimization techniques (a.k.a. lumping) after each composition. Formally, IMCs extend Continues Time Markov Chains by introducing action labels to transitions. I/O-IMCs extend IMCs by making the action transitions either direct or delayed events. A direct transition, denoted by a **x!**, is an output event, and a delayed transition, denoted by **x?**, is an input event. These I/O-IMCs allow one component to inform other components about its failure, so that they can react on it. The Arcade analysis method has been implemented in a toolset using CADP as a back-end [3].

IV. Arcade model of water distribution

To model the distribution part of a water treatment facility in Arcade, we create BCs for the valves and the tank. A valve can fail in two ways, it can either be stuck closed or open. In the following, we assume that a valve stuck open in the distribution process does not constitute a failure as it does not disrupt the distribution. Hence, a valve being stuck closed or open is modeled by two separate transitions. A tank can fail because of several reasons, it can rupture, be contaminated or leak. For simplicity, we assume in the following that a tank can only fail in one way. It is also assumed that the pipes do not fail and are therefore not modeled. For simplicity every BC has its own RU.

By composing the RUs with the corresponding BCs, we obtain a general I/O-IMC for a valve and one for the tank. Figure 3(a) shows the composed I/O-IMC model of a valve.



(a) Valve I/O-IMC (b) Tank I/O-IMC

Fig. 3. I/O-IMC of water treatment Arcade model

For every valve, an instance of this model with the values $\lambda_{valve-open} = \lambda_{valve-close} = 3/6000$ and $\mu_{valve} = 1$ is used. This corresponds to a valve breaking slowly and being repaired quickly. Figure 3(b) shows the composed I/O-IMC model of a tank. For this model, the values $\lambda_{tank} = 1/6000$ and $\mu_{tank} = 5/60$ are used. This corresponds to the tank failing more slowly than a valve, and also being repaired more slowly than a valve. The unit of time for the given rates is per hour. Note that these are made-up values that do not reflect reality.

From Figure 2 we extract a fault tree that describes the need for both water districts to receive water. This fault tree can be written down as follows:

$$\begin{aligned} System_{fail} \equiv\ & \\ & Valve4_{fail} \vee Valve5_{fail} \\ & \vee Valve6_{fail} \vee Tank_{fail} \\ & \vee (Valve1_{fail} \wedge Valve2_{fail}) \\ & \vee ((Valve1_{fail} \vee Valve2_{fail}) \wedge Valve3_{fail}). \end{aligned}$$

With the above I/O-IMCs and the specified fault tree Arcade can be used to create the minimized version of the underlying CTMC.

V. Dependability and survivability measures

Reliability is defined [7] as the probability of having no system failure within a certain mission time assuming that no component is repaired; availability is defined [1] as the probability of the system being in an operational state within a mission time assuming that components are repaired. In [4], Cloth et al. define survivability as the ability of a system to recover predefined service levels in a timely manner after the occurrence of disasters. Using CSL, this property can be expressed as a formula stating that any disaster is recoverable, that is

$$\textit{survivability} \equiv \textit{disaster} \Rightarrow \textit{recoverability}.$$

Recoverability for a given time bound *t* and probability bound *p* means that the system returns to predefined service levels before time *t* with at least probability *p*, formalized by the following CSL formula.

$$\textit{recoverability} \equiv \mathcal{P}_{\geq p}(\Diamond^{\leq t} \textit{service}).$$

Survivability analysis in Arcade. To compute survivability in Arcade the failure and operation states of the components

need to be identifiable in the composed system model. Without this information, it is impossible to use the CSL formulas to compute survivability. However, during the Arcade composing process the information about the failure of a specific component is lost. Only the operational status of the entire system is known, and not that of the specific component. To circumvent this problem two possible solution were considered: (i) No hiding: by not hiding transitions when parallel composing components, the transition information remains usable. However, this has a side effect on the size of the state space. (ii) Adding atomic properties (APs) to states: the states as used by CADP models have no APs, and CADP does not have the option to add them. An AP could be added to a state by creating a self-loop for the state indicating the AP.

The first option gave us a state space that could not be created in four days of calculation (The last count was approximately 5 million states and approximately 77 million transitions), whereas the second option results in a state space of only 35330 states and 405112 transitions. The size of the state space of option one makes computing any measures on it a time consuming task. Therefore, it was not used.

Arcade does not provide any means of checking models against CSL formulas. Hence, MRMC [5] is used to model check the underlying CTMC for survivability. However, in order to use MRMC, the Arcade model needs to be transformed in the following way: (i) The *fail* and *up* transitions that indicate the state of the entire system need to be hidden. (ii) All tau transitions present in the Arcade model are removed by using the CADP REDUCTOR tool. This does not affect the model, as the tau transitions are unimportant remnants of input and output transitions that were used for composing.

These steps reduce the size of the state space even more. The resulting MRMC model has 1458 states and 13122 transitions.

Survivability specifications The notion of survivability that we follow in this paper requires the definition of disasters and service levels as CSL formulas.

We define three different disasters that can possibly occur in the distribution process. The first disaster is valve4 failing, while all other components are still functional:

$$disaster\_1 \equiv (Valve4_{fail}).$$

The second disaster is a failing tank:

$$disaster\_2 \equiv (Tank_{fail}).$$

The third disaster is defined to be valve1 and valve3 failing:

$$disaster\_3 \equiv (Valve1_{fail} \wedge Valve3_{fail}).$$

The three service levels are defined as follows. In Service level one only district 1 receives water:

$$\begin{aligned} service\_level\_1 \quad \equiv \quad & (Valve1_{up} \wedge Valve4_{up}) \\ & \vee (Valve2_{up} \wedge Valve3_{up} \wedge Valve4_{up}). \end{aligned}$$

In Service level two only district 2 receives water:

$$\begin{aligned} service\_level\_2 \quad \equiv \quad & (Valve2_{up} \vee (Valve1_{up} \wedge Valve3_{up})) \\ & \wedge Valve5_{up} \wedge Valve6_{up} \wedge Tank_{up}. \end{aligned}$$

And in Service level three both districts receive water:

$$\begin{aligned} service\_level\_3 \quad \equiv \quad & Valve1_{up} \wedge Valve2_{up} \wedge Valve3_{up} \\ & \wedge Valve4_{up} \wedge Valve5_{up} \\ & \wedge Valve6_{up} \wedge Tank_{up}. \end{aligned}$$

VI. Dependability Evaluation

In this section, we present and discuss the resulting availability (A), reliability (R) and survivability (S) of the water distribution process. Figure 4 shows the availability of the distribution station over time. While at time zero the availability of the distribution process is 1 it decreases within the first 505 hours (approx. half a year) to 0.92, while the long-run availability is 0.84. Figure 5 illustrates the reliability of the distribution station over time. It can be seen that the model quickly becomes less reliable if no repairs are performed. After 1345 hours the system is only operational with probability 0.48. Using the different definitions of disasters and service

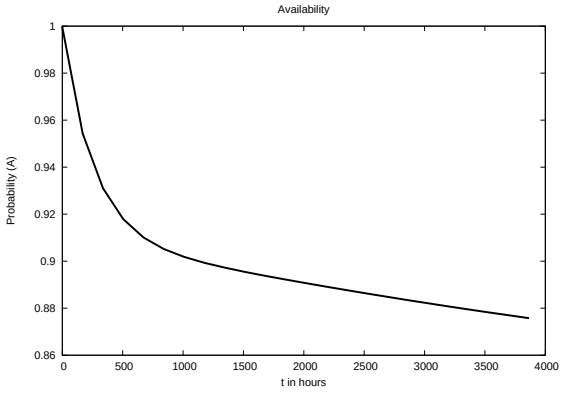


Fig. 4. Availability over time

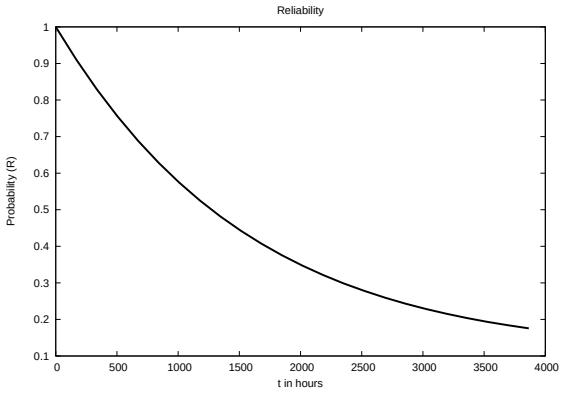


Fig. 5. Reliability over time

levels as described in Section V, we compute the survivability of the distribution station over time using MRMC. Figure 6 shows how fast the system recovers to *service_level_1* after the occurrence of one of the three different disasters. For *disaster_2* the probability to recover to *service_level_1* is 1, immediately (hence, this line does not show). This is because a failure of the tank does not influence the distribution to

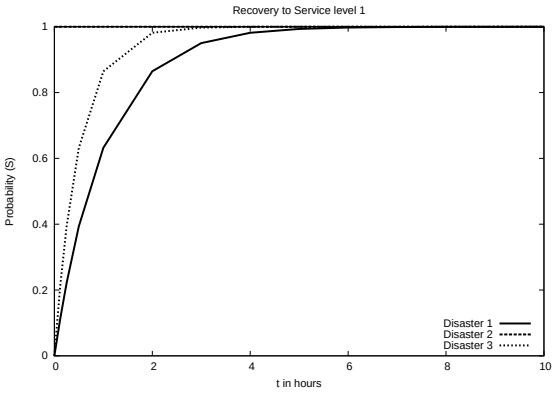


Fig. 6. Survivability over time (Service level 1)

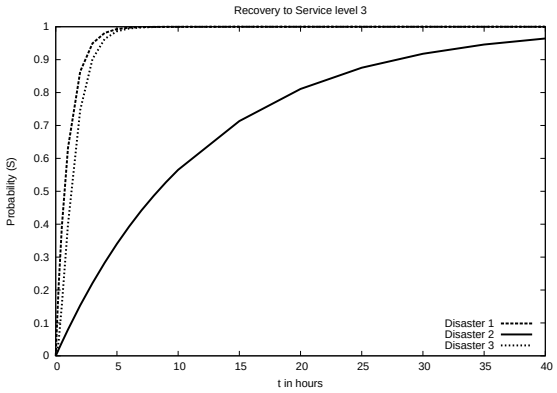


Fig. 8. Survivability over time (Service level 3)

water district 1. After the occurrence of *disaster_1*, i.e. the failure of valve4, the process is recovered with probability 1 after 17 hours. Whereas, after the failure of both valve3 and valve1, the process is fully recovered after 10 hours. This can be explained as follows. The repair of either valve3 or valve1 is sufficient to again distribute water to District 1. Hence, the probability of repair is higher than in the case of *disaster_1*. Figure 7 shows how the system recovers to *service_level_2*

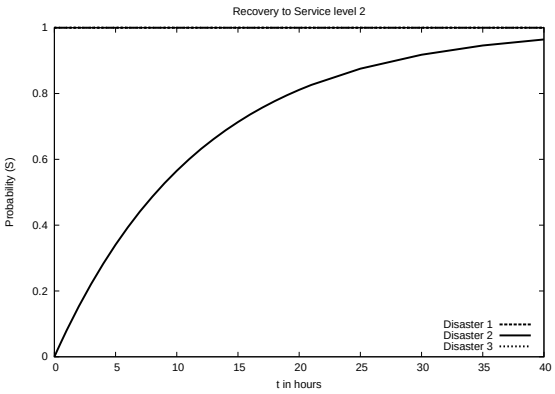


Fig. 7. Survivability over time (Service level 2)

given the occurrence of one of the three disasters. *Disaster_1* and *disaster_3* do not have any impact on the distribution to the second district, hence probability to recover from these disasters is 1, immediately. The recovery from a tank failure (*disaster_2*) is slow, the probability to reach *service_level_2* after 30 hours is only 0.92. This is due to the longer repair time for a tank. Note that the longer mean time to failure (MTTF) for a tank is not taken into account, as we do not model the failure explicitly, but assume that the failure has just happened. Figure 8 shows that to recover from *disaster_1* or from *disaster_3* (valve failures) to *service_level_3* (full service) with probability 1 only takes a few hours. Again, the recovery from a tank failure (*disaster_2*) takes much longer due to the lower repair rate.

VII. Conclusions

In this paper, we have presented our preliminary work on dependability and survivability modeling and analysis of the water distribution process in a water treatment facility. Using Arcade we have created a CTMC of the water distribution station, and directly computed availability and reliability. Furthermore, we have adapted the Arcade model to be model checked by MRMC for survivability. We conclude that the survivability model, resulting from Arcade is correct since MRMC produced exactly the same survivability measures for a manually created CTMC of the distribution station.

Future work includes a more comprehensive analysis of the water distribution station. In particular, we plan to compute quantitative properties. For instance, rather than computing the survivability as the probability to reach a predefined discrete service level, we plan to derive the expected service level (in terms of available water) after a disaster. On the implementation side, we plan to incorporate CSL model checking and survivability analysis within the Arcade toolset, so that future analysis becomes easier.

Acknowledgment. We thank Pepijn Crouzen for his help with Arcade and Matthias Kuntz for his valuable comments on the paper.

References

[1] A. Avizienis, J.-C Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1:11–33, 2004.

[2] H. Boudali, P. Crouzen, B. R. Haverkort, M. Kuntz, and M. Stoelinga. Architectural dependability evaluation with Arcade. In *Proceedings of the 38th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks*, pages 512–521. IEEE Computer Society Press, 2008.

[3] Construction and Analysis of Distributed Processes (CADP). http://www.inrialpes.fr/vasy/cadp.html.

[4] L. Cloth and B.R. Haverkort. Model checking for survivability! In *Proceedings of the 2nd Int. Conference on the Quantitative Evaluation of Systems*, pages 145–154. IEEE Computer Society Press, 2005.

[5] Markov Reward Model Checker (MRMC). http://www.mrmc-tool.org/.

[6] Rapport bescherming vitale infrastructuur. Technical report, Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, 2005.

[7] W. H. Sanders and L. M. Malhis. Dependability Evaluation Using Composed SAN-Based Reward Models. *Journal of Parallel and Distributed Computing 15*, pages 238–254, 1992.