

Exam - Friday 25. may 2002

SIE 5025 Pålitelige systemer *Dependable systems*

Solution to problems

Version 0.1; 13 May 2001; BEH

Task 1

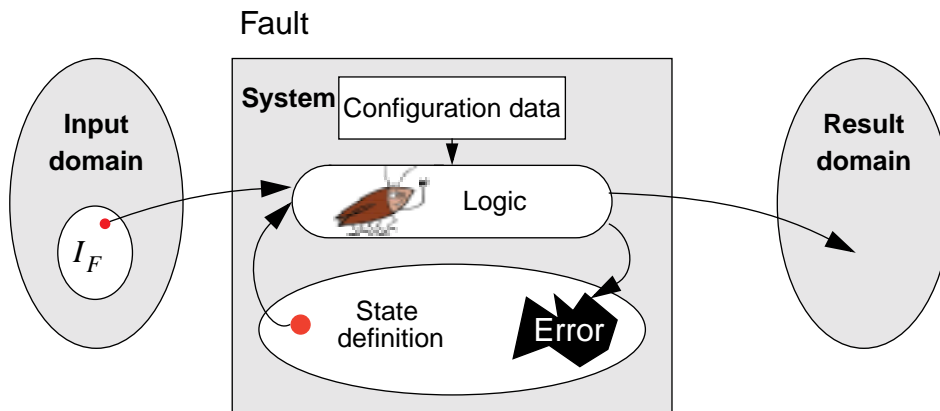
- a) **The direct method:** An empirical linear regression model is established between direct metrics of the code and the number of logical faults (later) found in the code during debugging, etc. This model is later used for prediction of the fault content of new code. In its simplest form just the number of lines of code is used as a metric. More elaborate version may use counts of various kind of statements.

Halsteads software science. uses the number of distinct and occurrences of operands and operators to predict various properties of the code, among them the Fault content. The “science” has adopted ideas from information theory and calculates/predicts the number of thought-steps necessary to produce the code. The fault content is found as the ratio between this quantity and the human failure intensity.

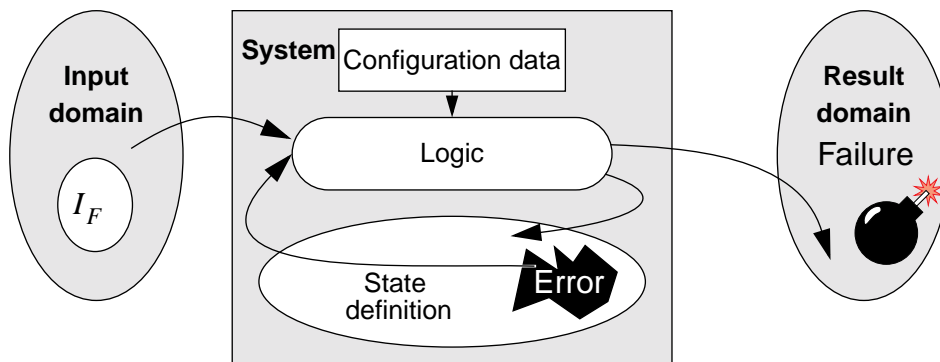
McCabes cyclomatic number. The control flow of the code is expressed as a directed graph. The complexity of the software is measured by the (cyclomatic) complexity of the graph. This complexity is empirically used to predict the fault content. $[v(G)=E-N+2F$; E is the number of edges, N is the number of nodes, and F is the number of entry points in the software. The number is the number independent cycles in a planar graph]

- b) <<Rewritten excerpt from lecture notes>> The software failure process of continuously operating software may be described by model (the Moore/Mealy like model discussed in the lecture notes and) illustrated in **Figur 1.1**. In this model, an input from the fault unveiling domain I_F will activate a logical fault in the software. Most likely this will first cause the faulty logic of the software to introduce an error in the state space of the software. (The fault activation may be conditioned by a specific (set of) internal states of the software, and hence, *it is the combination of the internal state and the input which causes the fault activation.*)

After a latency period this error may cause a failure of the software system. The failure may be triggered by an input outside I_F as illustrated in **Figur 1.1.b**. (Again we may have a *combination of the internal state and the input*)



a) Activation of a dormant logical fault

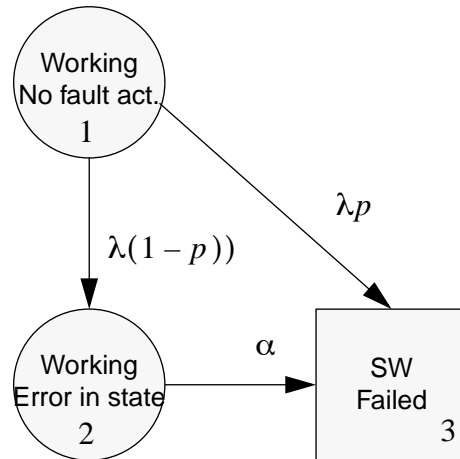


b) Software failure

Figur 1.1 Failure in the I/O-model extended to include state space

- c) Time from embedding of a fault into the software and until error occurrence is called **fault dormancy** and is described by the pdf $\lambda e^{-\lambda\tau}$.

Time from error occurrence until failure is called the **error latency** and is described by the pdf $p \cdot \delta(\tau) + (1 - p)\alpha e^{-\alpha\tau}$.



d)

The probability of being in state i at time t is denoted $P_i(t)$. The state vector of the system are defined as

$$\underline{P}(t) = \{P_1(t), P_2(t), P_3(t)\}^T.$$

The probability of being in the various states are determined by the set of linear differential equations with initial condition below.

$$\frac{d}{dt}\underline{P}(t) = \Lambda \underline{P}(t) \quad \text{where } \Lambda = \begin{bmatrix} -\lambda & 0 & 0 \\ \lambda(1-p) & -\alpha & 0 \\ \lambda p & \alpha & 0 \end{bmatrix}$$

$$\underline{P}(0) = \{1, 0, 0\}^T$$

The reliability function is the probability of being in an up-state. Hence,

$$R(t) = P_1(t) + P_2(t) = 1 - P_3(t)$$

e) We have that (if not remembered, the expression below is easily derived from the

$$\text{definition of failure rate} = \lim_{\Delta t \rightarrow 0} \frac{P(t < T_{FF} \leq t + \Delta t \mid T_F)}{\Delta t}.$$

$$\lambda(t) = \frac{f(t)}{R(t)} = \frac{-\frac{d}{dt}R(t)}{R(t)} = \frac{\lambda((\alpha - \lambda p)e^{-\lambda t} - \alpha(1-p)e^{-\alpha t})}{(\alpha - \lambda p)e^{-\lambda t} - \lambda(1-p)e^{-\alpha t}} \quad (1.1)$$

f) We have, when the system is stationary, that the failure intensity $z = 1/\text{MTBF}$. Since restarts are immediate and the system is as new after a restart, we have $\text{MTBF} = \text{MTFF}$. Hence,

$$z = (\text{MTFF})^{-1} = \left(\int_0^\infty R(t) dt \right)^{-1} = (\alpha - \lambda) \left(\int_0^\infty (\alpha - \lambda p)e^{-\lambda t} - \lambda(1-p)e^{-\alpha t} dt \right)^{-1} \quad (1.2)$$

$$= (\alpha - \lambda) \left(\frac{\alpha - \lambda p}{\lambda} - \frac{\lambda(1-p)}{\alpha} \right)^{-1} = \frac{\alpha \lambda}{\alpha + \lambda(1-p)}$$

(Alternative solution: The above answer may also be obtained from the state diagram by introducing a transition from state 3 to 1, find the intensity from the steady state solution of this and let the restart rate go to infinity to obtain the above.)

- g) It is seen that letting $t \rightarrow \infty$ in (1.1) yields an undefined expression. Using Hopitals rule does not resolve this problem. Hence, rearranging according to the tip and eliminates $e^{-\lambda\tau}$, i.e.,

$$\frac{\lambda((\alpha - \lambda p) - \alpha(1 - p)e^{(\lambda - \alpha)\tau})}{(\alpha - \lambda p) - \lambda(1 - p)e^{(\lambda - \alpha)\tau}} \quad (1.3)$$

Regards the case: $\alpha > \lambda$:

The last term in numerator and denominator will tend to zero and we obtain

$$\lim_{\tau \rightarrow \infty} \lambda(\tau) = \lambda$$

Regards the case: $\alpha < \lambda$:

The last term in numerator and denominator will tend towards infinity and dominate the first term which may be neglected. Reducing the remaining expression we obtain $\lim_{\tau \rightarrow \infty} \lambda(\tau) = \alpha$

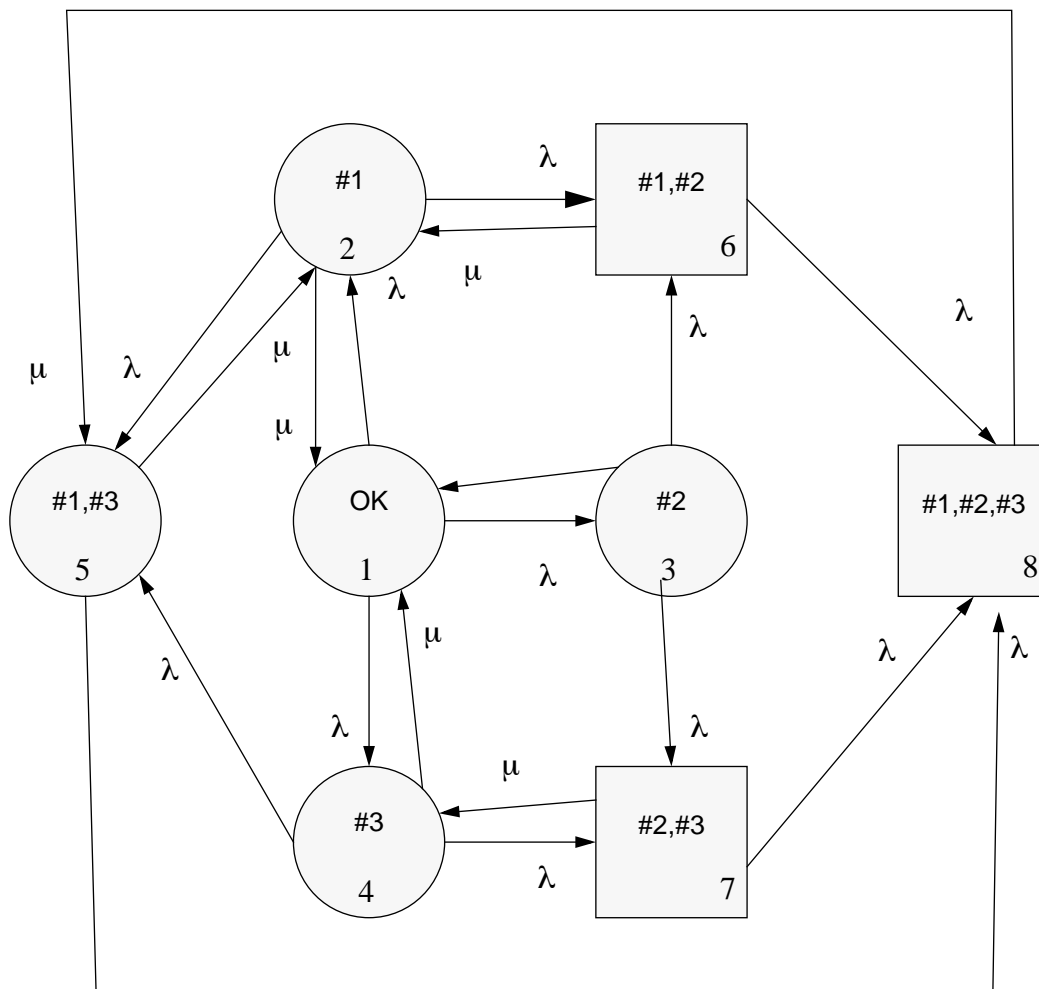
It is seen that in both cases, the limiting rate is different from the intensity. Both may be interpreted as events per second. The first is, however, the probability that the first failure will occur at the next instant, given that it has not occurred before, i.e. not for an infinitely long time. The latter is the probability the next failure will occur in the next instant.

The two cases shows that it is the slowest part in the failure process that will dominate the failure rate after an infinitely long time.

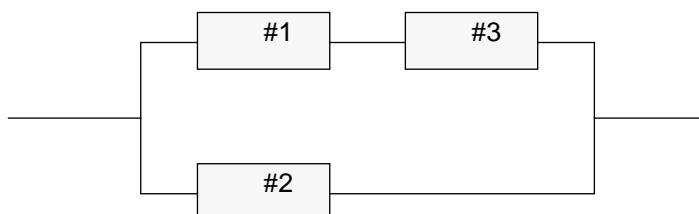
Task 2

- a) The two modifications are:
- The dependable registry should be used to obtain a reference to the group and a proxy for accessing the group.
 - For each operation on the server (invocation), it should be decided which invocation method (anycast or multicast) that should be used. This is decided according to the operation semantics, e.g. is the state of the server replica changed or not?, should the state of the replicas in the group be consistent?.
- b) **Anycast:** any single member of the group is accessed/invoked. Used in this case for getting an address since the replicas manages their own share of the address pool, i.e. an address from a randomly chosen part of the pool is returned.
- Multicast:** all members of the group is accessed/invoked. Used in this case refresh address leases since a) the client has no knowledge of which server that owns his address, b) all copies of the leased_address_database needs to be refreshed.

- c) To be on the safe side we should ensure that the processor and operating system had crash failure semantics. Jgroup is able to deal with timing failure semantics. This might be detected by jgroup as a temporary network partition, and state merge functionality must be implemented. Similarly jgroup can deal with omission failures as a network partition. jgroup is unable to deal value failure semantics.
- d) From the criteria defined in the task, processor #2 should always be repaired first. This yields the following state diagram, where the mnemonic of the state symbol indicates the failed processors:



e)



The availability of one processor is $A = \frac{\text{MUT}}{\text{MUT} + \text{MDT}} = \frac{\mu}{\lambda + \mu}$ since $\text{MUT} = \lambda^{-1}$ and $\text{MDT} = \mu^{-1}$.

From the block diagram we have:

$$U_s = 1 - A_s = (1 - A)(1 - A^2) = \frac{\lambda^2(\lambda + 2\mu)}{(\lambda + \mu)^3} \quad (2.1)$$

f) Since all processors are independent we may successively use the relations:

General

$$(1 - A) = \frac{\text{MDT}}{\text{MTBF}} \quad \text{MUT} = A \text{MTBF} \quad (2.2)$$

Series of two identical elements:

$$A_s = A^2 \quad \text{MUT}_s = 1 / \left(\frac{1}{\text{MUT}} + \frac{1}{\text{MUT}} \right) = \frac{1}{2} \text{MUT} = \frac{1}{2\lambda} \quad (2.3)$$

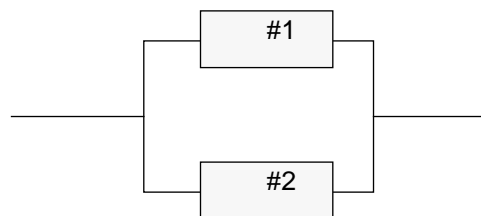
Parallel of two elements

$$(1 - A_p) = (1 - A_1)(1 - A_2) \quad \text{MDT}_p = 1 / \left(\frac{1}{\text{MDT}_1} + \frac{1}{\text{MDT}_2} \right)$$

Applied to the server subsystem - this yields.

Subsystem	Availability	MDT	MTBF
#i	$\frac{\mu}{\lambda + \mu}$	$\frac{1}{\mu}$	$\frac{1}{\lambda} + \frac{1}{\mu}$
#1,#3	$\frac{\mu^2}{(\lambda + \mu)^2}$	$\frac{\lambda + 2\mu}{2\mu^2}$	$\frac{(\lambda + \mu)^2}{2\lambda\mu^2}$
(#1,#3) #2	$\frac{\mu(\lambda^2 + 3\lambda\mu + \mu^2)}{(\lambda + \mu)^3}$	$\frac{\lambda + 2\mu}{\mu(\lambda + 4\mu)}$	$\frac{(\lambda + \mu)^3}{\lambda^2\mu(\lambda + 4\mu)}$

g) After the reconfiguration, the block diagram of the server subsystem becomes



With fewer processors that may fail, and bring the system down, the availability is improved and the unavailability is reduced. In this case, the unavailability becomes

$$U_{s^*} = 1 - A_{s^*} = (1 - A)^2 = \frac{\lambda^2}{(\lambda + \mu)^2} \quad (2.4)$$

and hence the ratio

$$\frac{U_{s^*}}{U_s} = \frac{(\lambda + \mu)}{(\lambda + 2\mu)} \quad (2.5)$$

(An alternative and faster way to the same result is to use

$$\frac{U_{s^*}}{U_s} = \frac{(1 - A)^2}{(1 - A)(1 - A^2)} = \frac{1}{1 + A} \quad (2.6)$$

It is seen that the ratio is $\frac{1}{2} < \frac{U_{s^*}}{U_s} < 1$.

If $\lambda \ll \mu$ we see that

$$\frac{U_{s^*}}{U_s} \approx \frac{1}{2} \text{ or more precisely } \frac{U_{s^*}}{U_s} = \frac{1}{2} + \frac{\lambda}{2(\lambda + 2\mu)} \approx \frac{1}{2} + \frac{\lambda}{4\mu} \approx \frac{1}{2}. \quad (2.7)$$

(Last expansion is not required at the examn).

When $\lambda \gg \mu$, $A \approx 0$ the the node #2 is down most of the time anyhow. The reduction in failure rate by using just node #1 for the other replicas does not matter.

When $\lambda \ll \mu$, $A \approx 1$, the failure rate and unavailability of the node(s) hosting replicas R_1, S_2 is almost halved, and hence, the probability of simultaneous failure of the two diagram branches/halves of the system are approx. the half.