Catherine Durso
106 John Greene Hall
303 871 3598 (office), 303 765 4335 (home)
cdurso@cs.du.edu
http://web.cs.du.edu/~cdurso, follow link

Texts: Probability and Statistics with Reliability, Queuing, and Computer Science Applications, ed. 2
Kishor S. Trivedi, available from Amazon, Fex.

This book is dense. We'll work to keep the details from running away with us. More basic, less applied versions of much of this material is available from
Basic Probability Theory, Ash
A First Course in Probability, Ross

Quantitative System Performance, Lazowska, Zahorian, Graham, Sevcik

This book is out of print but is available as a free download from
http://www.cs.washington.edu/homes/lazowska/qsp

It lays out basic techniques for simple, quick, rough analyses of computer systems, together with some extension techniques that allow for more detailed analyses.

Don't be alarmed by the quaint systems in the case studies. The fundamentals presented here haven't changed.

Keep an eye on how these early researchers pieced together obvious facts to get useful tools. Note also that some strong assumptions and powerful theory are incorporated without explanation. We'll discuss these issues after reviewing the techniques.

Grades:

problem sets: 10% each for 4 problem sets
in-class midterm: 30%
take home final: 30%

Homework may be done individually or in groups of up to three students. You may get help from me and any other credited, ethically sound source.

Final exam: format to be determined.

Prerequisites for the cause are primarily mathematical - basic calculus and basic linear algebra are required to follow the presentation.

Basic Topics of the Course

Computer Systems Analysis using Queuing Networks

typically

estimate performance parameters,
throughput, residence time

from input parameters
workload type / intensity
CPU requirements / speed
IO requirements / speed
memory requirements / speed

useful during
design

acquisition

adaptation (upgrades and changing workloads)

offers
middle path between gut feeling and experiment

strengths: quick, cheap, often informative

limitations: approximate, restricted to information about means, not distributions, some techniques assume relationships that may not hold in practice.

Probability Theory
background for refined analysis of queues and queue networks

background for simulation

applications in reliability and availability calculations

deeper and more flexible understanding of queuing theory

statistical analysis: assessing validity of probability models

Quantitative System Performance

Ch. 1 & Ch. 2 are orientation and of historical interest.

Queuing Network Modeling

"All models are wrong. Some models are useful."

Model system in
- design phase
- purchasing
- upgrading
- forecasting response to changed workload.

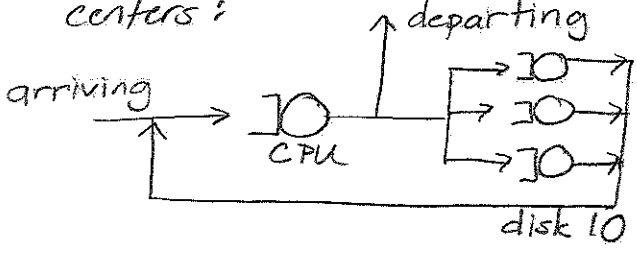
Model provides a useful abstraction allowing study of system performance, esp. response time on typical workloads.

Queuing Network Model

Computer System modeled as a network of service centers with queues (lines/pools) of jobs awaiting service.

Single service center notation:

Typical network w. multiple service centers:



(Note queuing theory is also used to analyze manufacturing processes, customer service, traffic flows, etc.)

Input: work load intensity (average) as a rate or count

service demand per transaction

Output: utilization: proportion of time device is active or average number in service/unit time

residence time: average time at node, wait time + service time

queue length average

throughput: jobs/sec, f. ex. average

Note the techniques in this book deal with averages not distributions. What are the implications of this?

We also make assumptions about arrival rates and service times that make the sequence of service center visits unimportant.

Modeling can be refined with multiple customer classes having different workload intensities and different service demands.

These techniques are adapted to batch, terminal, and transaction classes.

batch: constant number of jobs in system, all ready for service

terminal: constant number of clients in the system, some ready, some 'thinking'

transaction: variable number of clients in the system

Broad Outline of Modeling Processes

Frame questions.
Construct the abstract model.
Collect data to parametrize the model (input data).
Check the model (if possible) on an existing or known system.
Use the model to estimate desired quantities.
Assess accuracy of predictions where possible.

These steps are not entirely sequential. The steps interact.

Examples of modeling processes?

Examples of interactions?

Standard terms: Construction, Validation, Projection, Verification.

Specific issues for Queuing System Analysis

Models are comparatively simple and abstract. Modeler must capture relevant features of system. Emphasize primary effects.

ex. For a CPU upgrade, the primary effect is a decrease in CPU service demands.
Secondary effects may include changes in paging and swapping.

Quality and availability of measurements:
Profiling tools are better for some things - utilization - than others - tracking specific tasks. Observation alters the system.

Evaluation of Queuing Models
- bounds
- estimates

Strengths: Simpler than simulation. Doesn't require materials, in contrast to experimentation. Relatively simple inputs. With parametrization of models, simulations, & experiments imprecise, the accuracy of these models may be best suited to the quality of the data.

By analogy, if you're counting seconds between lightning and thunder by saying "1 elephant, 2 elephant...", you might as well use 345 m/sec as the speed of sound, rather than calculating the local speed of sound to cm/sec using temperature and humidity.

Simplicity facilitates exploring sensitivity to range of options – parameter values, model structure.

Useful in hybrid models combining analytic, benchmark, and simulation methods. Ex. Use simulation to evaluate submodels not fitting analytic framework of a queuing system.

Limitations: These techniques may assume a separable framework, defined later, but basically requiring behavior at each service center to be independent of populations at other service centers.

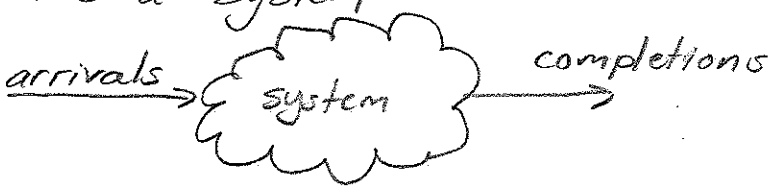
Ex. Having job use first available of interchangeable devices does not fit the framework.

(Adjust by hiding aggregate performance of the subsystem by other methods)

Ch. 3

Definitions of basic quantities, Little's Law, Forced Flow Law, Flow Balance Assumption, examples, summary tables

Observe a system



record

T: length of time of observation

A: number of arrivals observed

C: number of completions observed

If the system is a single resource, you may have access to

B: amount of time resource was busy

calculate

$\lambda: \frac{A}{T}$, arrival rate

$X: \frac{C}{T}$, completion rate, throughput

$U: \frac{B}{T}$, utilization, proportion of time the resource is busy

$S: \frac{B}{C}$, service requirement, average use of resource per completion

derive Utilization Law:

$$\frac{B}{T} = \frac{C}{T} \cdot \frac{B}{C} \quad \text{or}$$

$$U = X \cdot S$$

Utilization = Throughput · Service requirement

This is a special case of Little's Law

$$N = XR$$

Average number of requests in system =

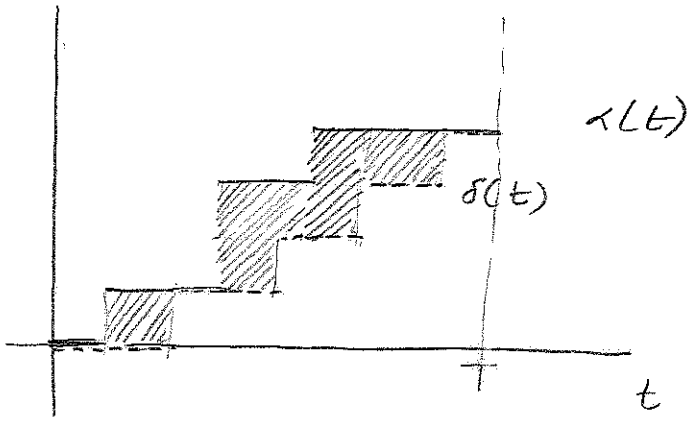
Average Throughput · Average time in system per request

for the case of a system without a separate queue: $N = \frac{B}{T}$, $S = R$

To understand Little's Law, define N, X, and R carefully.

Observe a system over a time interval $[0, T)$. Define $\alpha(t)$ = number of requests that arrived in $[0, t)$

Define $\delta(t)$ = number of departures in $[0, t)$



Set W equal to the area of the shaded region. Then W is request time units, say request minutes, in the system

Define

$N = \frac{W}{T}$ average number of requests in system

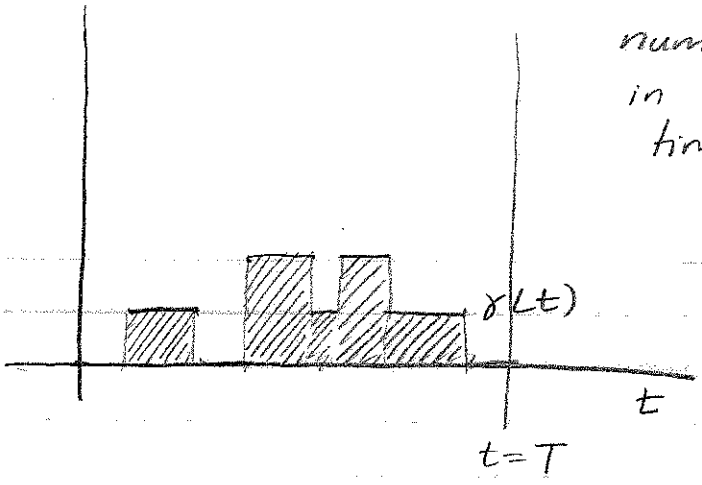
$R = \frac{W}{C}$ average residence time per request

$X = \frac{C}{T}$ throughput

$$\frac{W}{T} = \frac{C}{T} \cdot \frac{W}{C} \quad \text{so} \quad N = XR$$

For this to match our intuition, the number of requests in the system should be 0 at $t=0$ and $t=T$. If it is small compared to $\alpha(T)$ and $\delta(T)$, that will suffice.

Alternatively, we could plot $w(t) = \alpha(t) - \delta(t)$, so $w(t)$ is the number of requests in the system at time t.



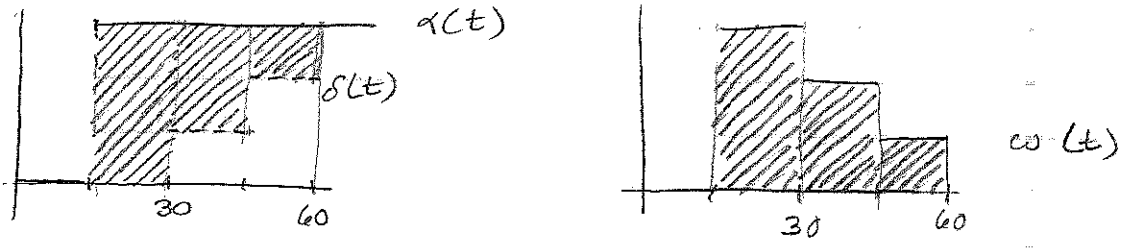
Then the area under $w(t)$ is W.

Little's Law in the gym:

Clients use stationary bike for 15 minutes

The observation period is 60 minutes

Case 1: 3 people arrive at $t = 15$



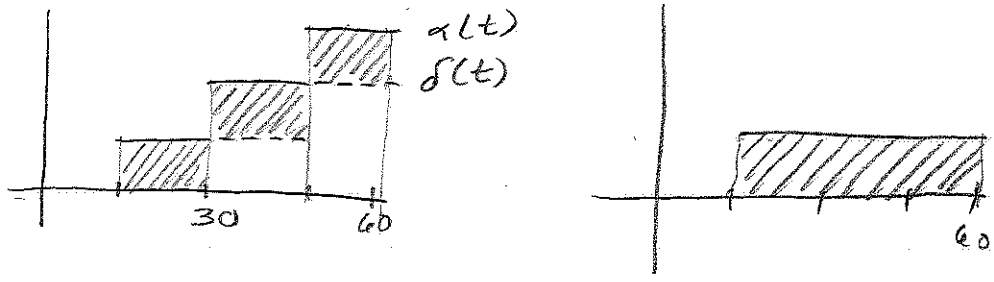
W = person minutes in the system $= 3 \cdot 15 + 2 \cdot 15 + 1 \cdot 15 = 90$

$N = \frac{W}{T} = \frac{90}{60} = 3/2$

R = average residence time $= \frac{W}{C} = \frac{90}{3} = 30$

$X = \frac{C}{T} = \frac{3}{60}$; Little's $\frac{3}{2} = \frac{3}{60} \cdot 30$

Case 1: one person arrives at $t=0$, one at $t=15$, and 1 at $t=45$

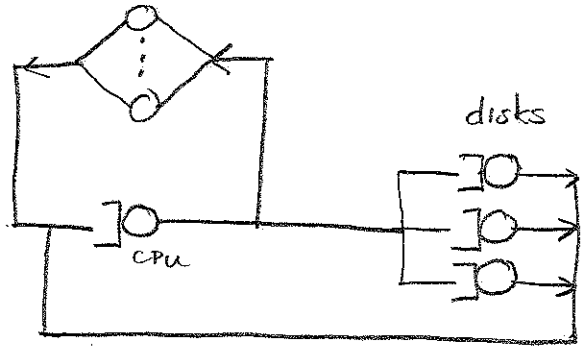


$W = 3 \cdot 15 = 45$

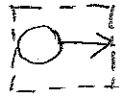
$N = \frac{45}{60} = \frac{3}{4}$, $R = \frac{45}{3} = 15$, $X = \frac{3}{60}$

$\frac{3}{4} = \frac{3}{60} \cdot \frac{45}{3}$ ✓

Time Share Example from text: Little's Law at 4 levels



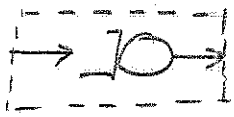
level 1: Single disk, not including queue: Suppose 40 requests/sec and average request takes .0225 sec.



$X = 40$, $S = .0225$

$N = U = XS = 40 \cdot .0225 = .9$, 90%

level 2: single disk and queue

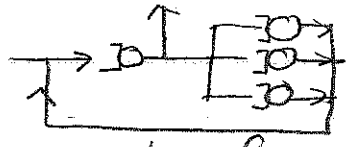


Still using 40 requests/sec and service time = .0225 sec, suppose also that the average number of requests total in queue and in service is 4.

$N = 4$, $X = 40$, so $N = XR$ implies

$$R = \frac{N}{X} = \frac{4}{40} = .1$$

level 3: central subsystem



N: population = average number of users with requests pending, ie not 'thinking', say $N = .75$ requests

X: average rate at which requests flow from users to the central subsystem, and from the central subsystem to the users (what assumption does this capture?)

say $X = .5$ requests/sec

$$R = \frac{N}{X} = \frac{.75}{.5} = 15 \text{ sec}$$

level 4 Entire system

N = number of users, say 10 (constant)

Suppose the central subsystem response time average remains 15 sec. and that users' work locally, on average, for 5 sec. before generating a request.

$Z = 5$

$R_{sys.} = R_{central} + Z = 15 + 5 = 20$

$$X = \frac{N}{R} = \frac{10}{20} = .5$$

Rewrite this as $X = \frac{N}{R_{central} + Z}$ or

$$R = NX - Z$$

to get the Response Time Law.

You may have noticed that we aren't solving the really interesting problem here: Given $N=10$, $Z=5$, and the service demands on the CPU and disks, we'd like system throughput and response time, say. But our gym example shows that details of arrival spacing affect this, so we need some assumptions and patience (ch. 5, 6, too). These exercises are useful for converting measurements to parameters.

Note that, due to numerical analysis issues, if the think time in a terminal workload is large compared to the central subsystem response time, a small variation in throughput data can cause a large percentage variation in the calculated subsystem response time.

Case 1 $N=64$, $Z=30$, $X=2$

$$R = \frac{N}{X} - Z = \frac{64}{2} - 30 = 2$$

Case 2 $N=64$, $Z=30$, $X=1.9$

$$R = \frac{64}{1.9} - 30 \approx 3.7$$

A difference of 5% in throughput results in an $\approx$ 85% difference in R.

Takeaway: Throughput is a more stable measure of performance than response time in these circumstances.

Forced Flow Law

Given a resource k in a system, a system level request may visit k repeatedly. The corresponding notation follows.

V_k : average number of visits to k by 1 request

C_k : number of completed visits to k in time T

C : number of system level completions

The forced flow law says that each service center has the number of completed visits required by the number of system level completions

$$C_k = V_k C$$

In terms of rates

$$X_k = V_k X \qquad \left(\text{i.e. } \frac{C_k}{T} = V_k \frac{C}{T} \right)$$

Correct response: Well, duh.

This says things like

"If system level requests require 3 disk reads on average and the system processes 5 requests/sec, then it must process $3 \cdot 5 = 15$ disk reads/sec. on average."

"The system is serving 150 customers/sec and doing 750 searches/sec. Thus each customer requires $\frac{X_k}{X} = \frac{750}{150} = 5 = V_k$ searches."

Application of identities and laws

Given 25 terminals,
18 sec. average think time,
20 visits to a particular disk per interaction,
30% utilization of that disk, and
.025 sec. average service requirement per visit at the disk,

find the average response time of the system.

The indicated formula is

$$R = \frac{N}{X} - Z \text{ , but we don't have } X.$$

But $X = \frac{X_d}{V_d}$, $V_d = 20$ and

$$U_d = X_d S_d \quad \text{or}$$

$.3 = X_d (.025)$, ie $X_d = 12$

$$X = \frac{12}{20} = .6, \text{ so}$$

$$R = \frac{25}{.6} - 16 \approx 23.7$$

More terminology

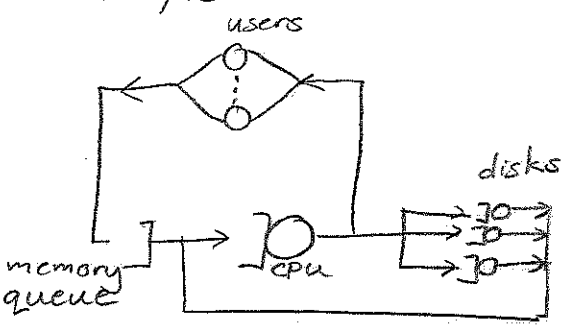
D_k = service demand per job from resource k = $V_k S_k$

$$D = \sum_k D_k$$

so

$$U_k = X_k S_k = X D_k$$

Example



In this model, jobs queue for memory.

Given

23 users
average response time perceived by users: 30 sec
timesharing throughput: 0.45 interactions/sec
average number of timesharing requests occupying memory: 1.9
average CPU service requirement/interaction: .63 sec.

- what is the average think time of users?
- On average, how many users were queuing for memory or occupying memory, totaled?
- On average, how many were queuing for memory?
- On average, how much time elapses between acquisition of memory and completion of the interaction?

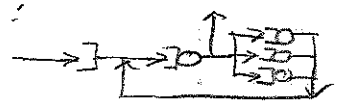
Think time: At the system level,
$N=23$, $R=30$, $X=.45$, so

$R = \frac{N}{X} - Z$ gives $Z = \frac{N}{X} - R$

$= \frac{23}{.45} - 30 = 21.\bar{1}$ (≈ 21)

# of users awaiting or receiving service:

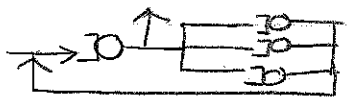
$N_{subsys} = XR = .45 \cdot 30 = 13.5$



average number queuing for memory

$13.5 - 1.9 = 11.6$

On average, time elapsed between acquisition of memory and completion of transaction



$N_{in\,mem} = X\, R_{in\,mem}, \quad 1.9 = .45\, R_{in\,mem}$

$R_{in\,mem} = \frac{1.9}{.45} = 4.\overline{2} \approx 4.2$

What is U_{cpu}?

$U_{cpu} = X \cdot D_{cpu} = .45 \cdot .63 = .2835$
$\approx .28$

Flow balance

If the system is not backing up an increasing pool of unfilled requests, arrivals = completions, $A = C$, so $A/T = C/T$, ie $\lambda = X$, arrival rate = throughput.

Example. Suppose you have three transaction classes, compilation, execution, and editing, with the following arrival rates and disk 1 service demand.

	arrival rate	disk 1 service demand
compilation	300/hour	1 sec.
execution	100/hour	5 sec.
editing	500/hour	.4 sec

What is the utilization of disk 1?

$U_k = X D_k$, and sum over the transaction classes. Note $X = \lambda$ for each transaction class.

Working in seconds,

$$\frac{300}{3600} \cdot 1 + \frac{100}{3600} \cdot 5 + \frac{500}{3600} \cdot 0.4 = .2\overline{7}$$

Facility in identifying and relating these quantities will come

See tables in text and handout

Ch. 4 Queuing Network Model Inputs and Outputs

Determine aspects of systems behavior. these techniques will and won't be able to represent. Extend the notations for input and output quantities.

We represent customers and service centers.

Customer description:

1) transaction workload: customers arrive, complete transactions, then leave. The workload is described by the arrival rate λ.
examples:

2) batch workload: batch workload has a fixed population. A customer completing service leaves and is instantaneously replaced by a customer needing service. The workload is described by the constant N, the size of the population receiving service.
examples:

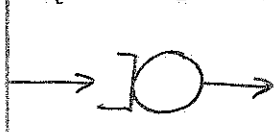
3. terminal workload: A fixed number of customers make occasional service demands N number of customers, Z average think time

examples:

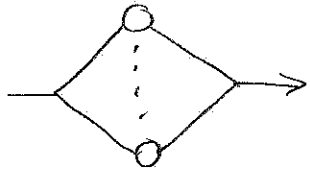
Case 1 is called an open workload.
Cases 2 and 3 are called closed workloads.
Note that for type 3, the population in the central subsystem varies.

Center Description

These models handle 2 basic center types: queuing centers, where customers compete for use of a resource, and delay centers, where customers are allocated a server at need.



queuing



Why have we represented customers in a terminal workload as a delay center?

Consider a vaccination program with effectively unlimited tables and pencils for each patient to complete vaccination forms, a small number of people administering vaccines, and a large supply of possible patients. How would you model this

Single class model: customers are random draws from a single pool. No scheduling discipline needs to be specified because customers are indistinguishable.

Multiple class model: customers divided into groups with possibly different average service demands and workloads. Transaction, batch, and timeshare may all be represented. Scheduling discipline may distinguish between classes.

To review the notation for single class models,

D_k: total amount of service required by a customer from center k (average)
V_k: average number of visits made to center k by a customer
S_k: average service time per visit to center k.

$$D_k = B_k/C = V_k S_k$$

The separable models we'll pursue here depend only on D_k, not S_k.

D: total service demand $= \sum_{k=1}^{K} D_k$

Model inputs: Customer description - λ, N, (N, Z)
Center description — queuing or delay
Service demands - $D_k = V_k S_k$

Model Outputs

System level: R - average system response time
X - system throughput
Q - average number in the system

Center level: U_k - utilization of center k
R_k - average residence time at center k, queuing time + service time
X_k - average throughput at center k, visits/unit time
Q_k - average queue length at center k

$$R = \sum R_k, \quad X_k = V_k X$$

batch: $Q = N$
transaction: $Q = XR$
terminal: $Q = N - XZ$ $(Q = XR, \; R = N/X - Z)$

Note these are all averages. Distributions of R or R_k, for example, might be of interest if one needs the probability that the response time is greater than some threshold.

Multiple class models; basically slap subscripts on inputs and outputs.

Say there are C classes (unrelated to C completions).

$$\vec{\lambda} = (\lambda_1, \lambda_2, \ldots \lambda_C), \quad \vec{N} = (N_1, N_2, \ldots N_C)$$

$$\vec{NZ} = ((N_1, Z_1), (N_2, Z_2), \ldots (N_C, Z_C))$$

or mixed

$$\vec{I} = ((N_1, N_1, Z_1, \text{or } \lambda_1), (N_2, (N_2, Z_2), \text{or } \lambda_2) \ldots (N_C, (N_C, Z_C), \text{or } \lambda_C))$$

Assume that the scheduling discipline is independent of class to preserve desirable properties of the model.

$$D_c = \sum_{k=1}^{K} D_{c,k}$$

$$U_k = \sum_{c=1}^{C} U_{c,k}$$

$$X = \sum_{k=1}^{K} X_k$$

$$Q_k = \sum_{c=1}^{C} Q_{c,k}$$

but

$$R = \sum_{c=1}^{C} \frac{R_c X_c}{X}, \qquad R_k = \sum_{c=1}^{C} \frac{R_{c,k} X_c}{X}$$

- reasonable, and necessary to preserve Little's law

The things we will not represent directly include

- simultaneous resource possession requirements (disk + controller + channel, for ex.)
- memory constraints: assume sufficient memory for N batch jobs, $\leq N$ terminal requests
- blocking: the state of one resource is assumed not to affect another
- process creation and synchronization
- high service time variability: use multiple classes
- priority scheduling
- response time distributions

Why does it still work?

Missing information may be represented indirectly.

In studies of modifications, proposed changes may not affect non represented properties

Extensions are available.