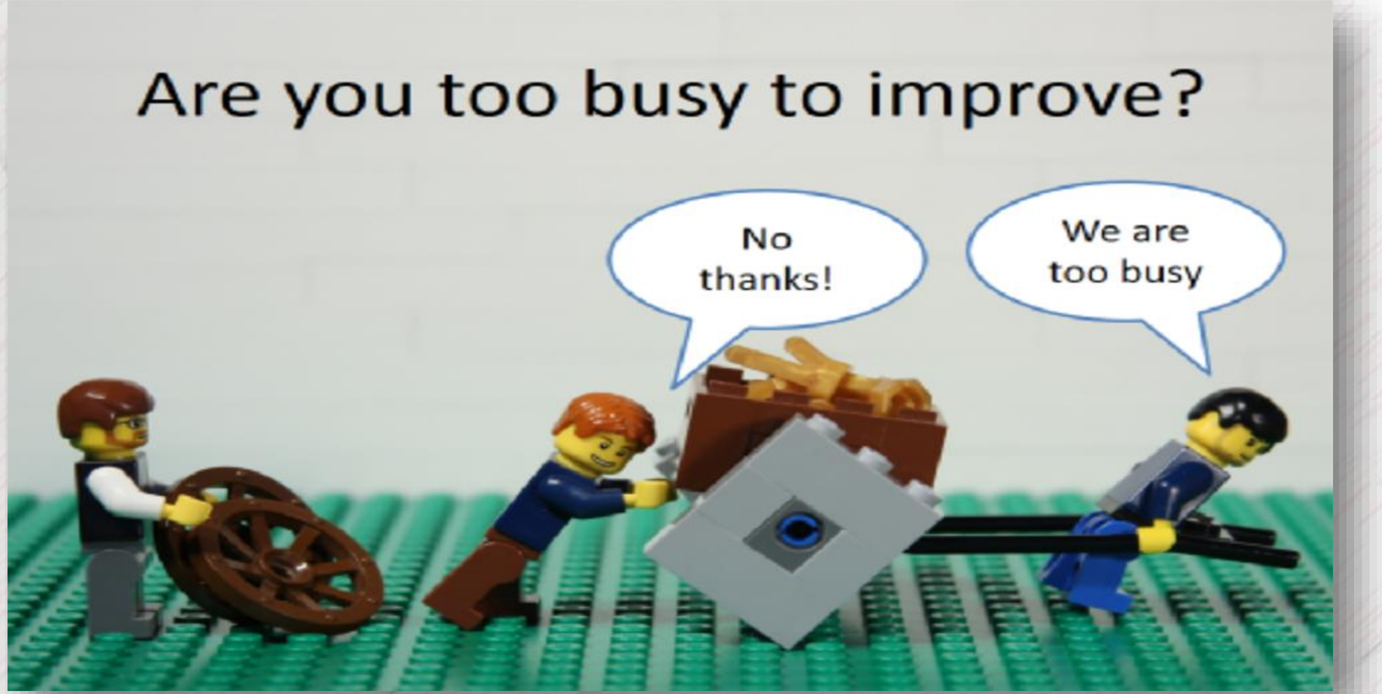
Analyzing Specifications, writing an optimized set of test cases

Best Testing
The best are getting better

Alon Linetzki
alonl@sigist.org.il

TAPOST

Latvia, Oct 2015



Alon Linetzki

- Coaching, consulting and training worldwide for the last 32 years in IT – development, testing, quality assurance
- Interested in how teams work efficiently and effectively and how can teams improve…
- Likes coaching and sharing knowledge.
- Keen on people and ideas

- Certified Scrum Master
- Certified System Analyst
- ISTQB Advanced Level – Test Manager and Test Analyst
- ISO-9000 Lead Assessor
- TMMi Assessor
- Certified Mobile Tester (CMAP)
- Certified Communication Skills coach
- MBA
- Ms.c Statistics & Criminology

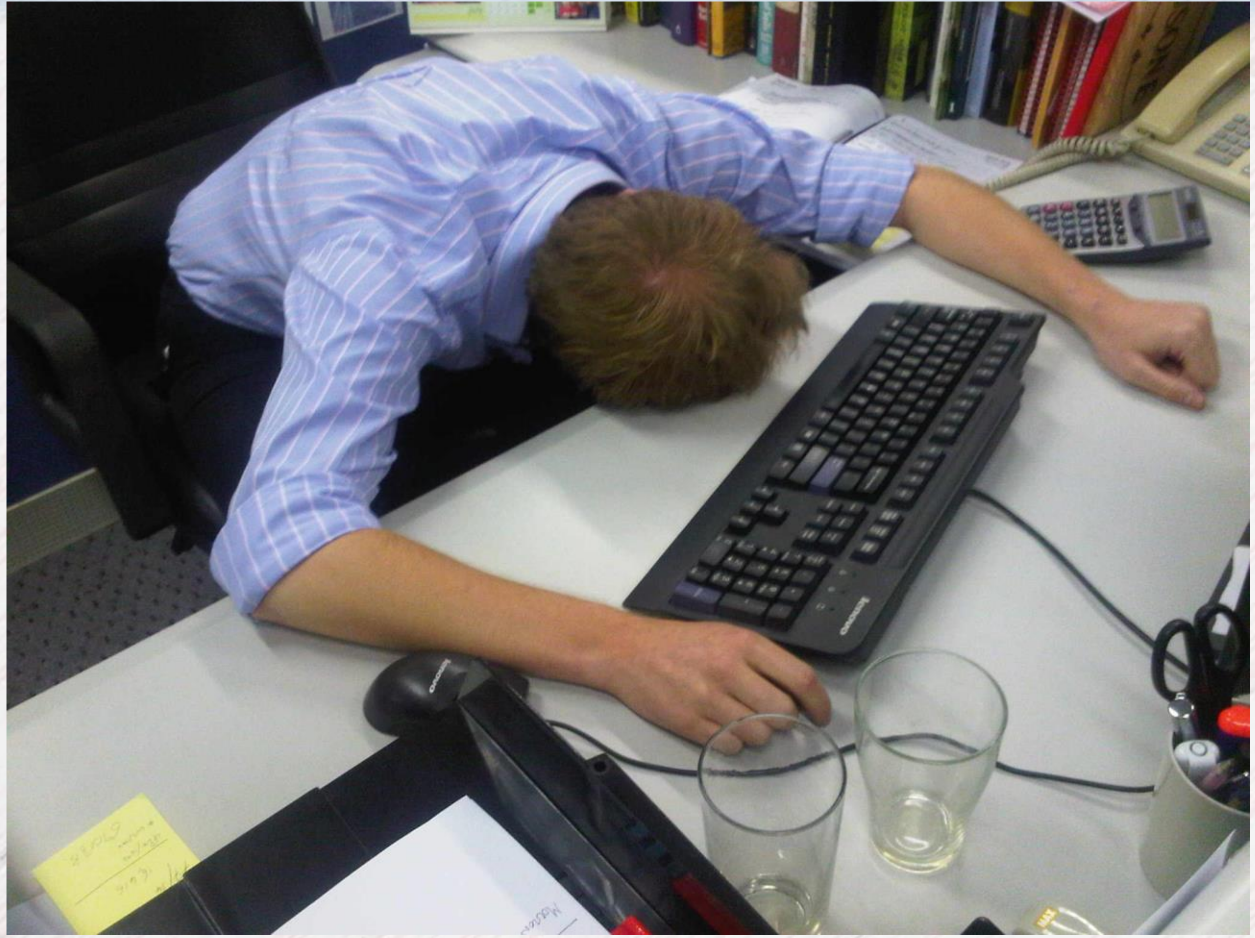
Agenda

- Motivation
- Suggested analysis & optimization framework
- Case Study
- Summary

You have received a specification document - and then What?

Analyze Specifications - But…

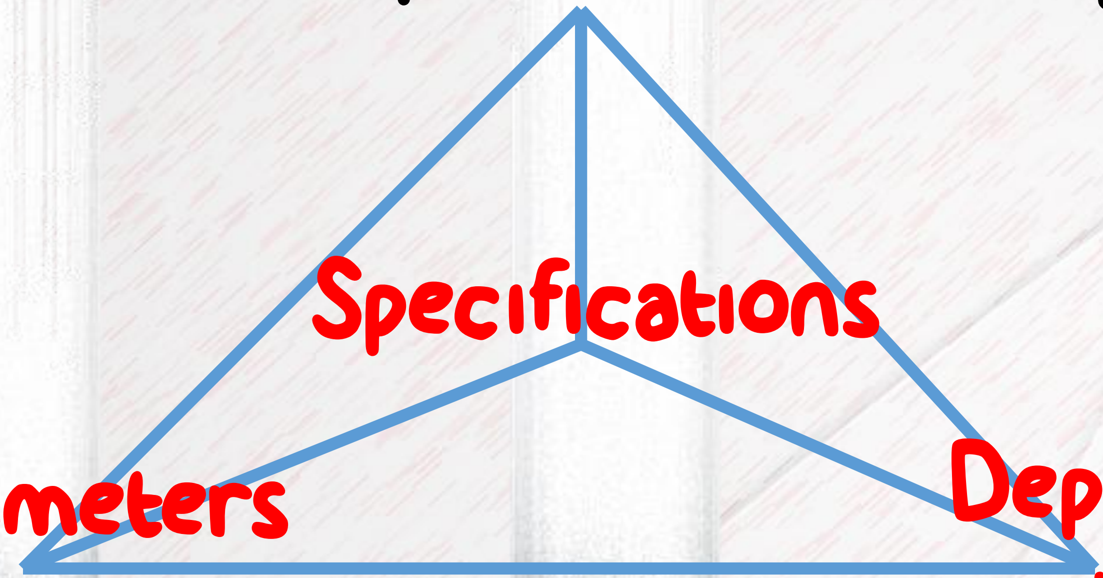
Exhaustive Testing is not possible!



So, how should we analyze the specs', in an effective and efficient way, and produce the best set of test cases?

What are the risks involved?

The process pillars...



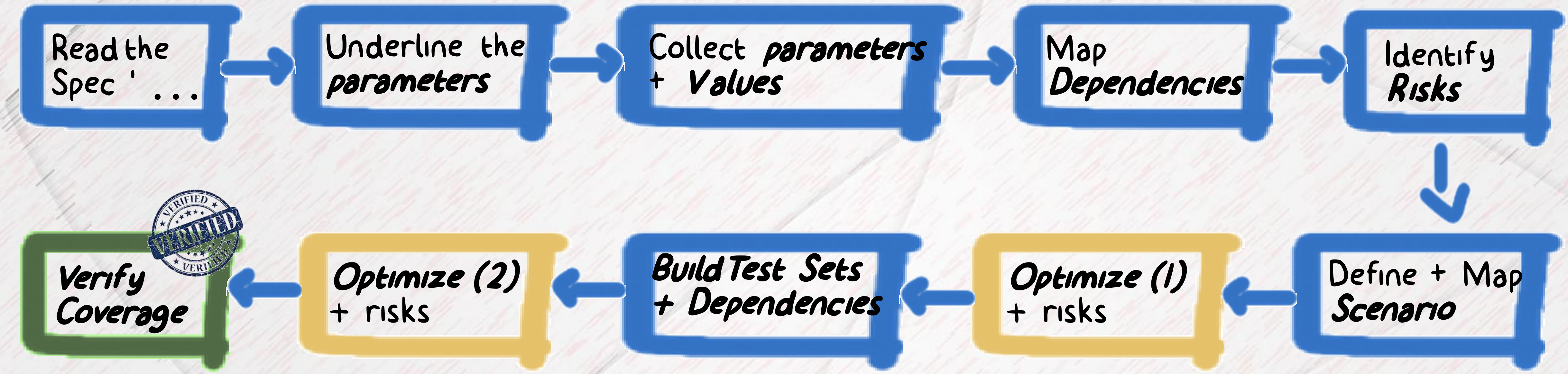
Optimization (using tools)

Specifications

Parameters + Values

Dependencies + Risks

Suggested Analysis + optimization framework



Read the Spec' ...

Underline the *parameters*

Collect *parameters* + *Values*

Map *Dependencies*

Identify *Risks*

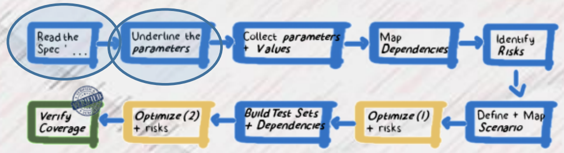
Define + Map *Scenario*

Optimize (1) + risks

Build Test Sets + Dependencies

Optimize (2) + risks

Verify Coverage



Read the Spec' ...

Underline the parameters

Collect parameters + Values

Map Dependencies

Identify Risks

Define + Map Scenario

Optimize (1) + risks

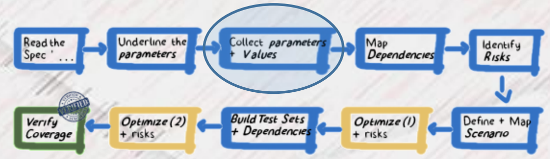
Build Test Sets + Dependencies

Optimize (2) + risks

Verify Coverage

Read the Specification text

- Specifications can be written in a: document, an email, a piece of paper…
- **Read** the specifications carefully
- **Write** down your **notes**, **questions** (unanswered), and **open issues**
- Meet Product and R&D and **close those gaps**
- Read the specification again
- **Mark the Parameters**…coming next…



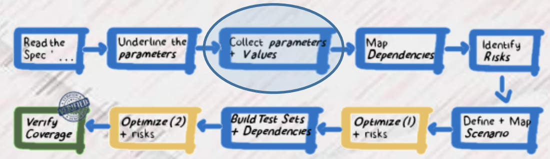
Collecting Parameters

- ***Parameters*** are fields, or changing set of values
- By choosing one value over another, we are changing the way the flow happens in the system, and usually having a different user experience and information
  - GUI
  - Non-GUI

- Examples:
  - Weight
  - Measuring units
  - Temperature
  - Customer type
  - Number of passengers
  - etc.

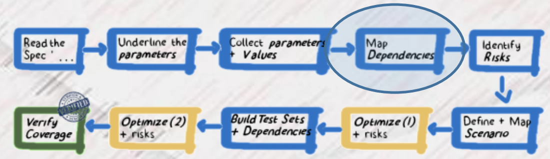
Some parameters are left outside:

- Customer ID – received from the system
- Device Name – coming from outside system
- Etc.



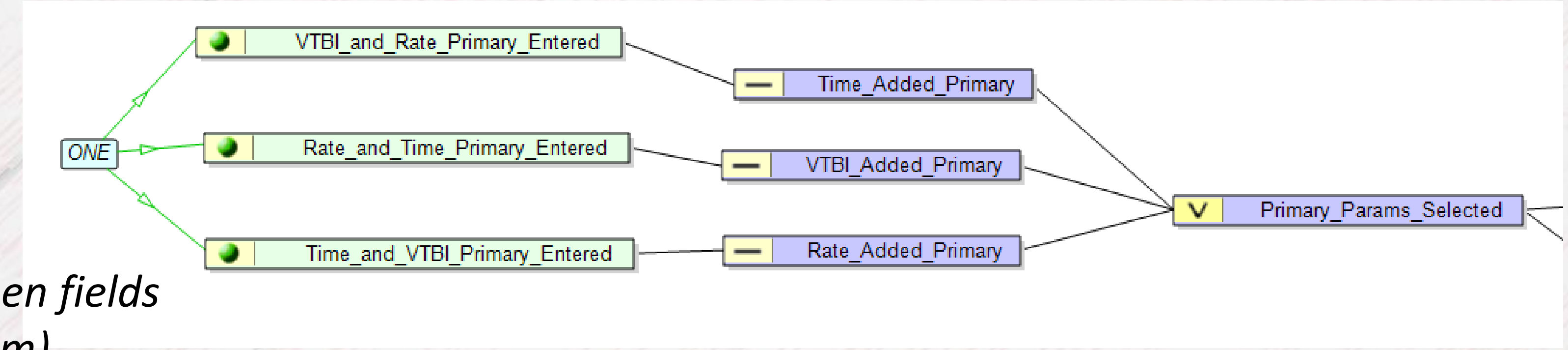
Identifying Values

- ***Values*** (of Parameters), are the different choices one can make in this parameter field.
- Examples:
  - Weight Unit – Kg/Grams
  - Distance Unit – Km, Meters, CM
  - Customer type – temp, member, VIP
  - Subscriber type – Regular, Alumni, member, etc.
  - Temperature – F, C
  - Number of passengers – 1, 2, 3, etc.
  - etc.

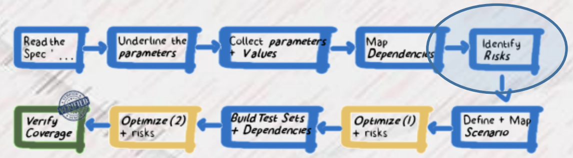


Map Dependencies

- Dependencies are ***Relations*** of values based on other values.
- If a parameter value has a ***logic*** connecting it to another parameter value, that is probably a dependency
- Examples:
  - Seating class → class type (economy, business, first)
  - From date <= To Date
  - Customer type (preferred guest/member) → room above 10th floor + access to VIP lounge
  - Etc.


ONE
VTBI_and_Rate_Primary_Entered
Rate_and_Time_Primary_Entered
Time_and_VTBI_Primary_Entered
Time_Added_Primary
VTBI_Added_Primary
Rate_Added_Primary
V
Primary_Params_Selected

Modeling relations between fields (MBT, cause-effect diagram)



Read the Spec' ...

Underline the parameters

Collect parameters + Values

Map Dependencies

Identify Risks

Define + Map Scenario

Optimize (1) + risks

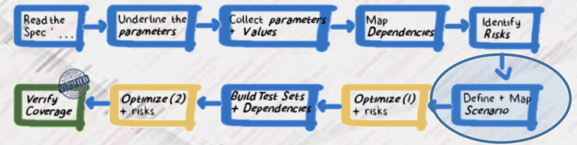
Build Test Sets + Dependencies

Optimize (2) + risks

Verify Coverage

Identifying Risks

- Risks can be many things…
- Examples:
  - *Critical & complex feature (new technology) ,*
  - *Low quality code received from unit testing* or *integration testing,*
  - Software Specifications *without enough level of details,*
  - *Safety related topics* in a medical device,
  - Etc.
- **Identify risks**, before constructing your tests – to have better focus in your test design, and know where you must cover with more tests.
- **Include these risks** when optimizing (1) (2)



Define your Scenarios Map

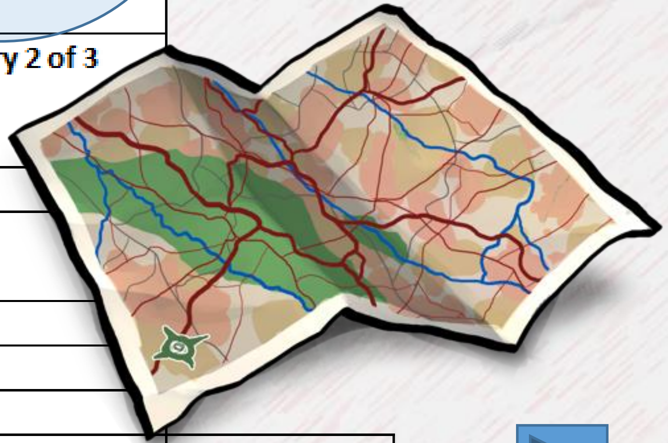
- Build your scenario map, using the parameters, values and dependencies

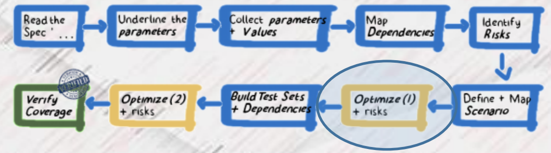
Continuous Infusion:

Dependencies

Parameters

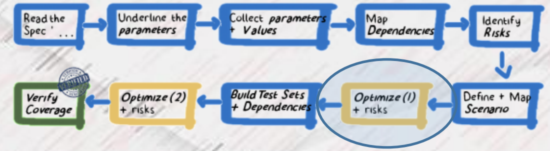
Dependency						only if Set secondary (technician) is Yes	only if Set secondary (technician) is Yes & Set Secondary is set	only if Set secondary (technician) is Yes & Set Secondary is set & mL is set	
Parameter	**mL**	**Dose units**	**2 of 3**	**Prime**	**Set Secondary (technician options)**	**Set Secondary**	**Secndry mL**	**Secndry 2 of 3**	
Values	mL/hr	grams/mL	Rate	Yes	Yes	Set Secondary	mL/hr	Rate	
	mL/min	mg/mL	VTBI	No	No	Start (no secondary)	mL/min	VTBI	
		mcg/mL	Time					Time	
		nanog/mL							
		mmol/nL							
	2	5	3	2	2	2	2	3	**1440**





Optimize (1)

- <u>Taking out parameters</u> to reduce combinations and the number of test cases.
- Take risks into considerations - NOT increasing the risk (based on coverage and type of product)
- Example:
  - Take *Prime* out – multiplications reduced in half (2 values),
  - Make sure to have a test case for *Prime*, to take care of the risk…



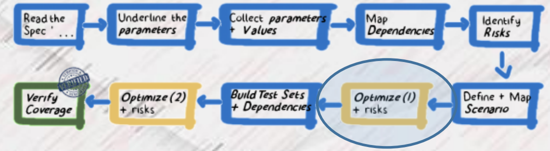
Read the Spec ' . . . → Underline the parameters → Collect parameters + Values → Map Dependencies → Identify Risks → Define + Map Scenario → Optimize (1) + risks → Build Test Sets + Dependencies → Optimize (2) + risks → Verify Coverage

Optimize (1)

- Taking out values (based on dependencies or just selecting 'least commonly used') – that reduces combinations and the number of test cases
- Make sure you are NOT increasing the risk

- Example:
  - Take out *least commonly used dosing units* (from ~37 units).
  - Select 3-5 most commonly used – ask product/clinical professionals to support your selection.
  - Multiplications reduced by much – it is ~15% of the original number of test cases

KEEP
CALM
AND
OPTIMIZE



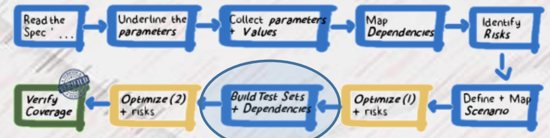
Optimize (1)

(1) Take *Prime* out
(2) Take *Dosing Units* out
(3) Merge "set secondary"

Continuous Infusion:

Dependency						only if Set secondary (technician) is Yes	only if Set secondary (technician) is Yes & Set Secondary is set	only if Set secondary (technician) is Yes & Set Secondary is set & mL is set	
Parameter	**mL**	**Dose units**	**2 of 3**	**Prime**	**Set Secondary (technician options)**	**Set Secondary**	**Secndry mL**	**Secndry 2 of 3**	
Values	mL/hr	grams/mL	Rate	Yes	Yes	Set Secondary	mL/hr	Rate	
	mL/min	mg/mL	VTBI	No	No	Start (no secondary)	mL/min	VTBI	
		mcg/mL	Time					Time	
		nanog/mL							
		mmol/mL							
	2	5	3	2	2	2	2	3	1440
Optimze	2	3	3	1	2	2	2	3	432
					can turn into 2 options only...				
	2	3	3	1	2		2	3	216

Build test combinations + Injected dependencies



Read the Spec ' …

Underline the parameters

Collect parameters + Values

Map Dependencies

Identify Risks

Define + Map Scenario

Optimize (1) + risks

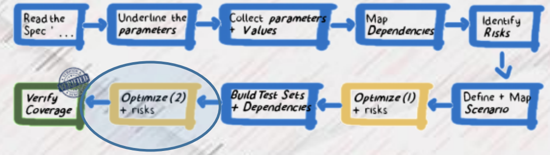
Build Test Sets + Dependencies

Optimize (2) + risks

Verify Coverage

- Create your ‘all-so-far’ combination matrix including: parameters, values, dependencies

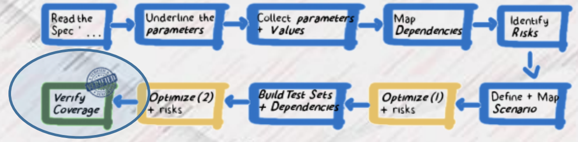
	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11
mL	mL/hr	mL/hr	mL/hr	mL/hr	mL/hr	mL/hr	mL/hr	mL/hr	mL/hr	…	…
Does Units	grams/mL	grams/mL	grams/mL	grams/mL	grams/mL	grams/mL	grams/mL	grams/mL	grams/mL	…	…
2 of 3	Rate+VTBI	Rate+Time	VTBI+Time	Rate+VTBI	Rate+Time	VTBI+Time	Rate+VTBI	Rate+Time	VTBI+Time	…	…
Prime	No	No	No	Yes	Yes	Yes	No	No	No	…	…
Set Secondary (T	No	No	No	No	No	No	Yes	Yes	Yes	…	…
Set Secondary	N/A	N/A	N/A	N/A	N/A	N/A	Set Secondary	Set Secondary	Set Secondary	…	…
Secondary mL	N/A	N/A	N/A	N/A	N/A	N/A	mL/hr	mL/hr	mL/hr	…	…
Secondary 2 of 3	N/A	N/A	N/A	N/A	N/A	N/A	Rate+VTBI	Rate+Time	VTBI+Time	…	…



Optimize (2)

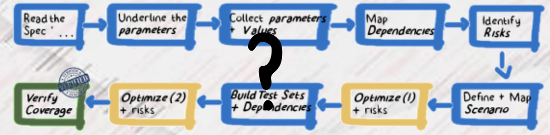
- Use *Pairwise, cause-effect graphing, Classification-Tree Analysis* or other optimization analysis methods and tools to reduce from here on, or stop and automate…

Ctrl
control yourself
Alt
Look For Alternative Solutions
Del
Delete Situations that give you tension & negative energy



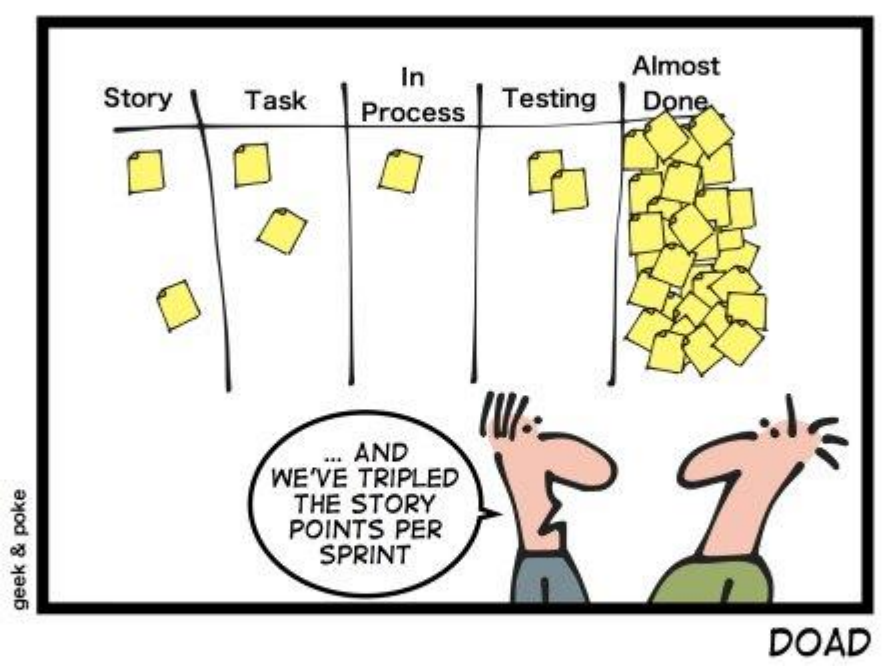
Verify Coverage

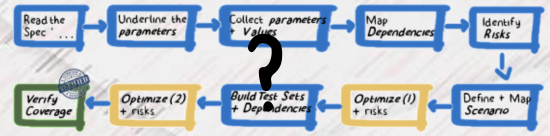
- Now you have a potential minimal set of test cases for your spec’s/problem.
- Verify with product – review test scenarios – to make sure the scenarios which they see day2day are present
- Verify risks have been translated into test cases in the right way and in the right place
- Make adjustments when necessary.



Done?

Are we 'done' or are we 'not done' – this is the question…





Read the Spec ' . . .

Underline the parameters

Collect parameters + Values

Map Dependencies

Identify Risks

Define + Map Scenario

Optimize (1) + risks

Build Test Sets + Dependencies

Optimize (2) + risks

Verify Coverage

Now . . . Automate!

- Run the set, make sure tests are effective and correct (no bugs).
- As this minimal set of test cases is to be run multiple times, on every build or when Verification efforts are in motion, **automating these can be very beneficial**.

A Case study

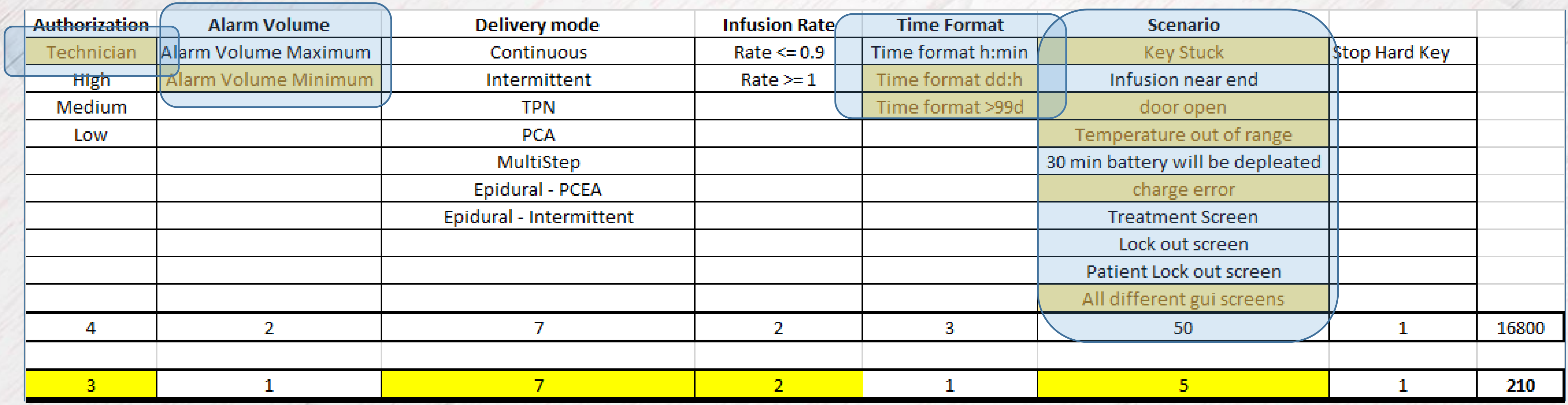
Case study – Blood infusion pump

- Complex and reach features
- Light weight
- Long battery duration
- Highly accurate
- Monitored by hospital control center
- Can be used at home
- 15 manual testers (8 new testers), 3 automation engineers
- High scope change rate
- Tight deadlines…

A Case study

- ‘***Stop Hard Key***’ functionality
- After a few hours of training session, with exercises, a senior tester analyzed the function below.

- Optimization decisions:
  - **Remove Technician**
  - **Remove alarm volume minimum factory default is maximum**
  - **Remove uncommon time formats**
  - **Remove rare events/scenarios**
  - **Remove uncommon GUI areas**



Authorization	Alarm Volume	Delivery mode	Infusion Rate	Time Format	Scenario		
Technician	Alarm Volume Maximum	Continuous	Rate <= 0.9	Time format h:min	Key Stuck	Stop Hard Key	
High	Alarm Volume Minimum	Intermittent	Rate >= 1	Time format dd:h	Infusion near end		
Medium		TPN		Time format >99d	door open		
Low		PCA			Temperature out of range		
		MultiStep			30 min battery will be depleated		
		Epidural - PCEA			charge error		
		Epidural - Intermittent			Treatment Screen		
					Lock out screen		
					Patient Lock out screen		
					All different gui screens		
4	2	7	2	3	50	1	16800
3	1	7	2	1	5	1	210

Further enhancement can be done by taking Boundary samples of the Infusion Rate: Rate = 0.9, <0.9, =1, >1

A Case Study

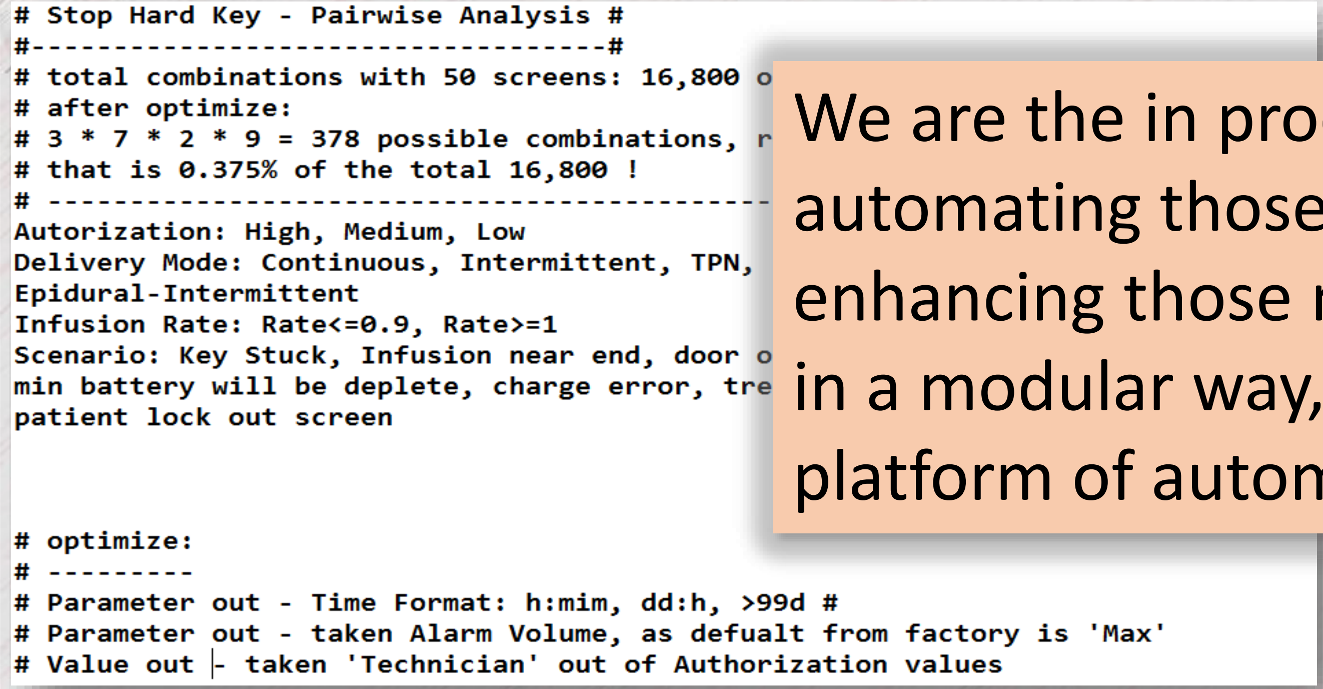
- Optimized below (including risk):
  - Taking Alarm Volume + Time Format parameters out
  - Taking colored cells out (not in scenario)
  - Taking ‘Technician’ value out
  - Taking 9 scenarios (GUI screens) into account – most commonly used

Reduced to 378 combinations

Authorization	Delivery mode	Infusion Rate	Scenario		
Technician	Continuous	Rate <= 0.9	Key Stuck	Stop Hard Key	
High	Intermittent	Rate >= 1	Infusion near end		
Medium	TPN		door open		
Low	PCA		Temperature out of range		
	MultiStep		30 min battery will be depleated		
	Epidural - PCEA		charge error		
	Epidural - Intermittent		Treatment Screen		
			Lock out screen		
			Patient Lock out screen		
			All different gui screens		
4	7	2	50	1	16800
3	7	2	9	1	378

A Case Study

- Optimized below:
  - Ran Pairwise (using PICT) on 378 tests…
  - Got 63 combinations, which are 0.4% from the original 16,800 combinations
  - Covering 100% of the pairs

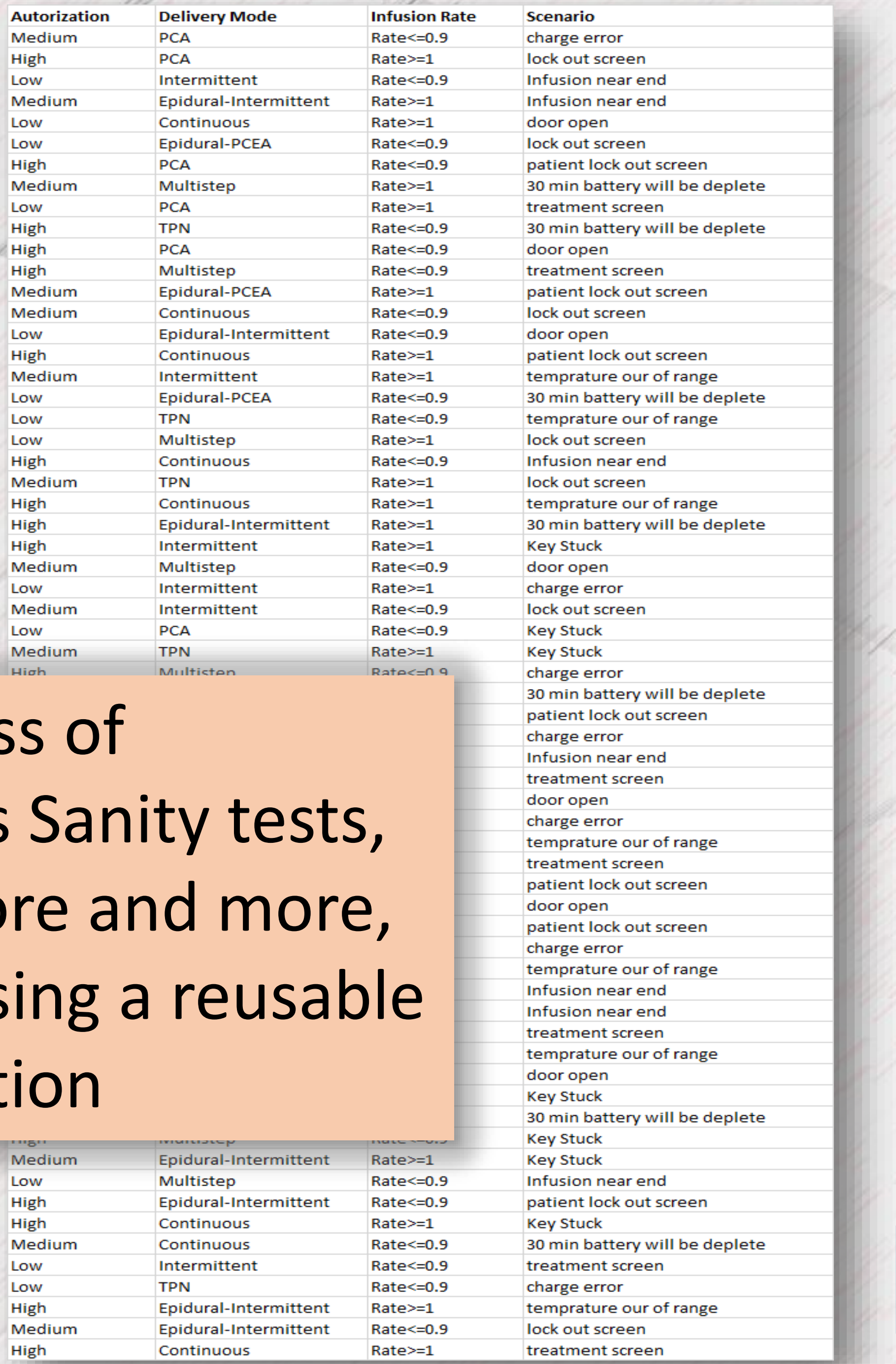


```
# Stop Hard Key - Pairwise Analysis #
#---------------------------------#
# total combinations with 50 screens: 16,800 o
# after optimize:
# 3 * 7 * 2 * 9 = 378 possible combinations, r
# that is 0.375% of the total 16,800 !
# ----------------------------------------
Autorization: High, Medium, Low
Delivery Mode: Continuous, Intermittent, TPN,
Epidural-Intermittent
Infusion Rate: Rate<=0.9, Rate>=1
Scenario: Key Stuck, Infusion near end, door o
min battery will be deplete, charge error, tre
patient lock out screen



# optimize:
# ---------
# Parameter out - Time Format: h:mim, dd:h, >99d #
# Parameter out - taken Alarm Volume, as defualt from factory is 'Max'
# Value out - taken 'Technician' out of Authorization values
```

We are the in process of automating those as Sanity tests, enhancing those more and more, in a modular way, using a reusable platform of automation



Autorization	Delivery Mode	Infusion Rate	Scenario
Medium	PCA	Rate<=0.9	charge error
High	PCA	Rate>=1	lock out screen
Low	Intermittent	Rate<=0.9	Infusion near end
Medium	Epidural-Intermittent	Rate>=1	Infusion near end
Low	Continuous	Rate>=1	door open
Low	Epidural-PCEA	Rate<=0.9	lock out screen
High	PCA	Rate<=0.9	patient lock out screen
Medium	Multistep	Rate>=1	30 min battery will be deplete
Low	PCA	Rate>=1	treatment screen
High	TPN	Rate<=0.9	30 min battery will be deplete
High	PCA	Rate<=0.9	door open
High	Multistep	Rate<=0.9	treatment screen
Medium	Epidural-PCEA	Rate>=1	patient lock out screen
Medium	Continuous	Rate<=0.9	lock out screen
Low	Epidural-Intermittent	Rate<=0.9	door open
High	Continuous	Rate>=1	patient lock out screen
Medium	Intermittent	Rate>=1	temprature our of range
Low	Epidural-PCEA	Rate<=0.9	30 min battery will be deplete
Low	TPN	Rate<=0.9	temprature our of range
Low	Multistep	Rate>=1	lock out screen
High	Continuous	Rate<=0.9	Infusion near end
Medium	TPN	Rate>=1	lock out screen
High	Continuous	Rate>=1	temprature our of range
High	Epidural-Intermittent	Rate>=1	30 min battery will be deplete
High	Intermittent	Rate>=1	Key Stuck
Medium	Multistep	Rate<=0.9	door open
Low	Intermittent	Rate>=1	charge error
Medium	Intermittent	Rate<=0.9	lock out screen
Low	PCA	Rate<=0.9	Key Stuck
Medium	TPN	Rate>=1	Key Stuck
High	Multistep	Rate<=0.9	charge error
[illegible]	[illegible]	[illegible]	30 min battery will be deplete
[illegible]	[illegible]	[illegible]	patient lock out screen
[illegible]	[illegible]	[illegible]	charge error
[illegible]	[illegible]	[illegible]	Infusion near end
[illegible]	[illegible]	[illegible]	treatment screen
[illegible]	[illegible]	[illegible]	door open
[illegible]	[illegible]	[illegible]	charge error
[illegible]	[illegible]	[illegible]	temprature our of range
[illegible]	[illegible]	[illegible]	treatment screen
[illegible]	[illegible]	[illegible]	patient lock out screen
[illegible]	[illegible]	[illegible]	door open
[illegible]	[illegible]	[illegible]	patient lock out screen
[illegible]	[illegible]	[illegible]	charge error
[illegible]	[illegible]	[illegible]	temprature our of range
[illegible]	[illegible]	[illegible]	Infusion near end
[illegible]	[illegible]	[illegible]	Infusion near end
[illegible]	[illegible]	[illegible]	treatment screen
[illegible]	[illegible]	[illegible]	temprature our of range
[illegible]	[illegible]	[illegible]	door open
[illegible]	[illegible]	[illegible]	Key Stuck
[illegible]	[illegible]	[illegible]	30 min battery will be deplete
High	Multistep	Rate<=0.9	Key Stuck
Medium	Epidural-Intermittent	Rate>=1	Key Stuck
Low	Multistep	Rate<=0.9	Infusion near end
High	Epidural-Intermittent	Rate<=0.9	patient lock out screen
High	Continuous	Rate>=1	Key Stuck
Medium	Continuous	Rate<=0.9	30 min battery will be deplete
Low	Intermittent	Rate<=0.9	treatment screen
Low	TPN	Rate<=0.9	charge error
High	Epidural-Intermittent	Rate>=1	temprature our of range
Medium	Epidural-Intermittent	Rate<=0.9	lock out screen
High	Continuous	Rate>=1	treatment screen

Summary

- Analyzing specifications with the aim of designing efficient and effective set of test cases – ‘min-max’
- Suggested framework process involving: parameters, values, risks, optimization, verification
- Using test design optimization tools – Classification-Tree Analysis & PICT
- Case study – How can we start?

Inspiration…

“The secret of change is to focus all your efforts, not on fighting the old, but on building the new.”

-- Socrates

Analyzing Specifications, writing an optimized set of test cases

BestTesting
The best are getting better

Alon Linetzki

Best – Testing

alonl@sigist.org.il

October 2015

Thank you!