

# Test Design

Robert Sabourin

President

AmiBug.Com, Inc.

Montreal, Canada

[rsabourin@amibug.com](mailto:rsabourin@amibug.com)

# Mind Maps

Now trying to absorb ideas, examples, techniques for Mind Maps

view of art

- more color
- more art
- more playful
- more enjoyable
- more creative?
- more memorable

Computer maps

- many different authors
- subjects
- purposes

- chaos theory
- nature
- beautiful

- imagination
- stimulating

- note taking
- note making
- presentations

- meetings
- minutes
- teaching

- learning
- and so on
- and so forth

Changed my Learning

The Mind Map Book  
by Tony Buzan

Usefulness



Thinking & Learning



Capacity  
great  
untapped  
all

Testimonial  
many  
diverse  
years

Personal  
I use  
try manual?

I like

I experience  
delight  
attraction  
growth

Scientific  
spotty  
weak  
trendy  
better may exist

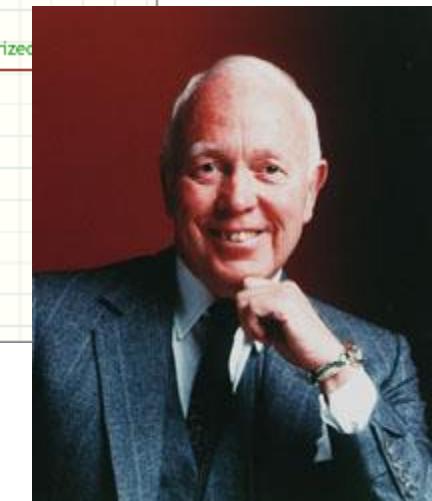
Author  
Academic  
Work

Psychology  
English  
Mathematics  
General Sciences

Journalist, editor  
Author



whole brain  
mnemonic  
relational

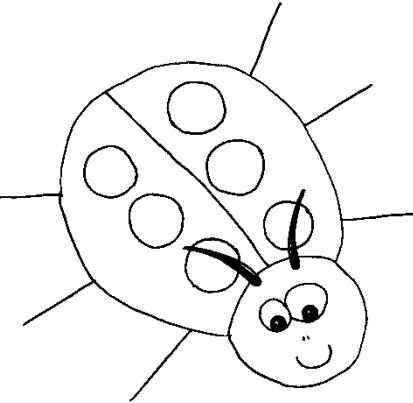


# Tony Buzan

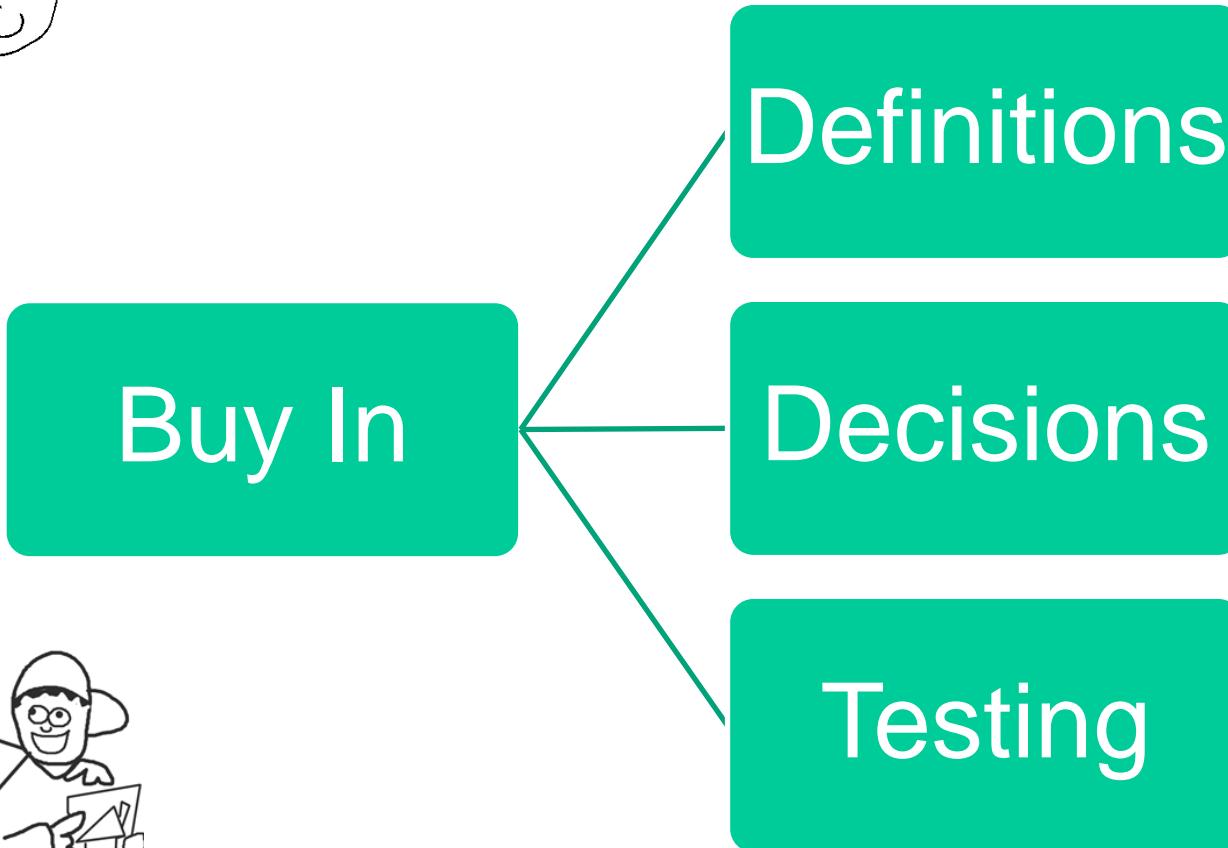
September 7, 2020

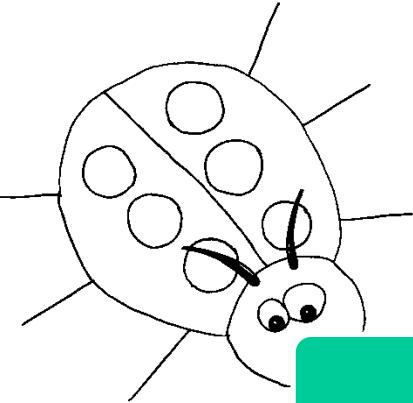
© Robert Sabourin, 2020

Slide 2



# Before Test Design





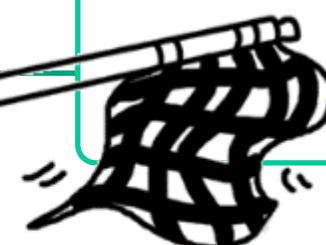
# Before Test Design

## Definitions

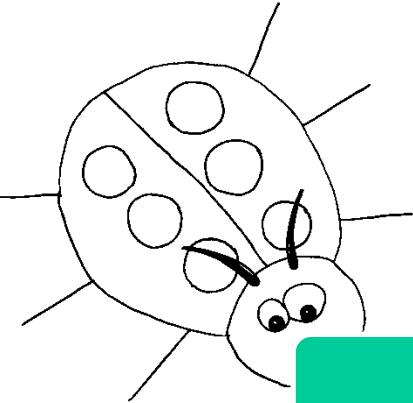


Quality

Bugs



Done



# Before Test Design

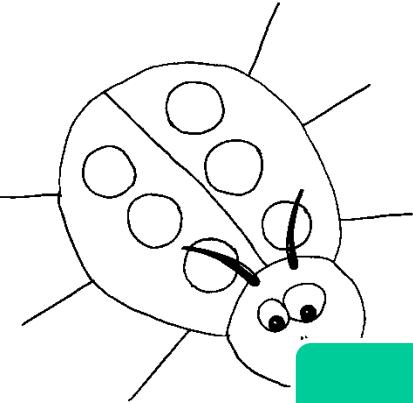


## Decisions

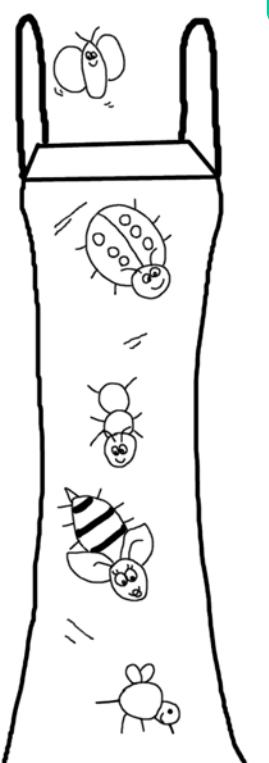
### Requirements

### Bugs

### Tests



# Before Test Design



## Testing

### Levels

### Scope

### Good enough



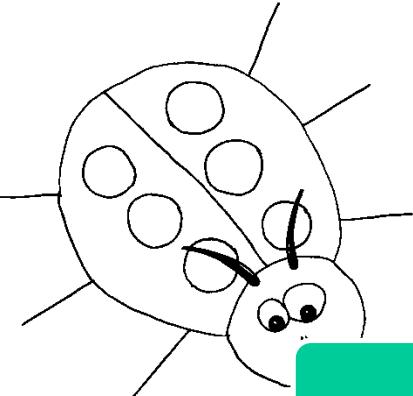
# Before Test Design

## Test Objective

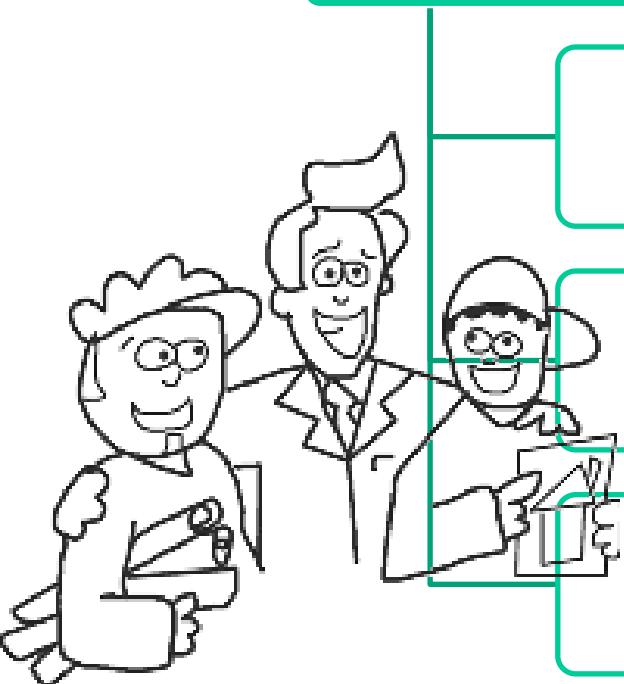
Identified

Important

Knowledge



# Value of Test Design

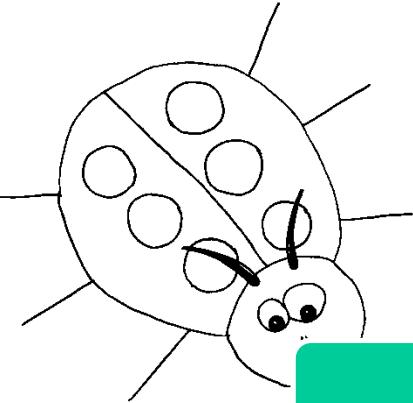


Visual design

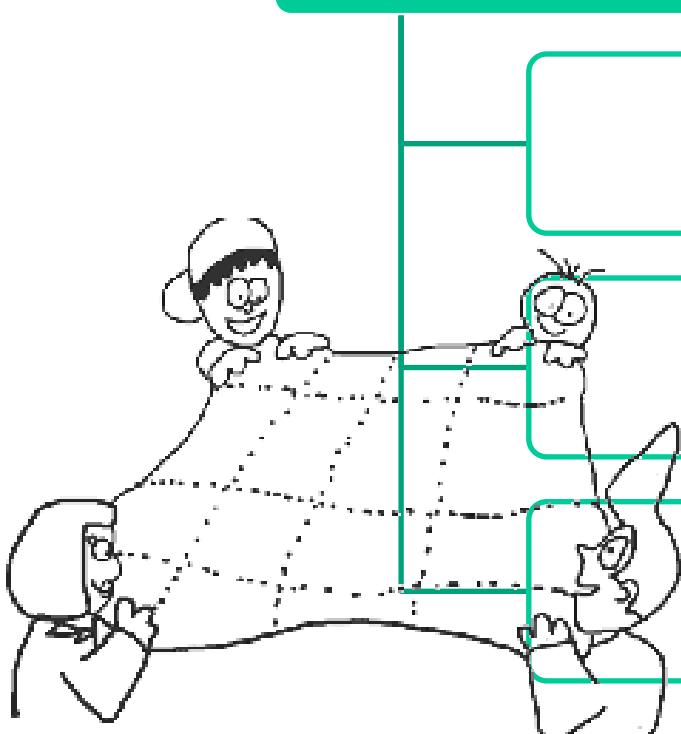
Blueprint

Focuses work

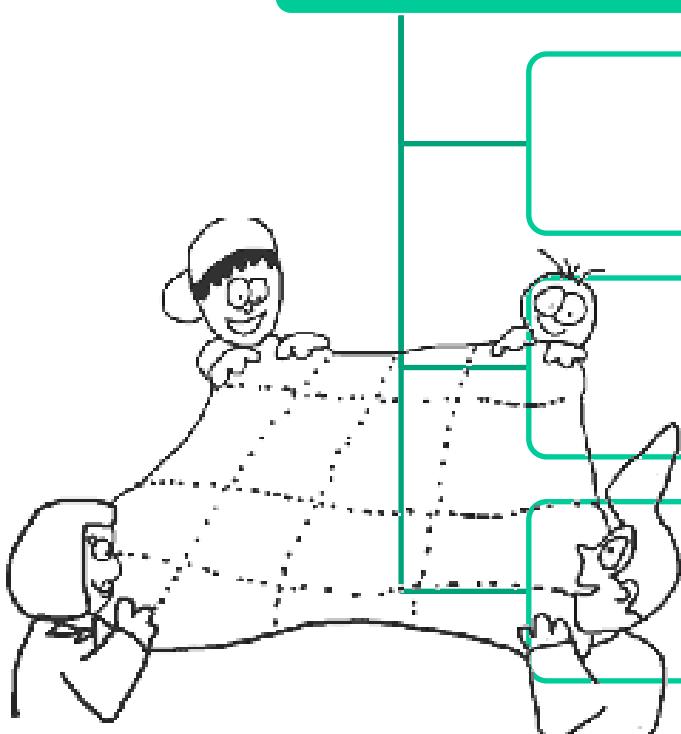
Reusable tool



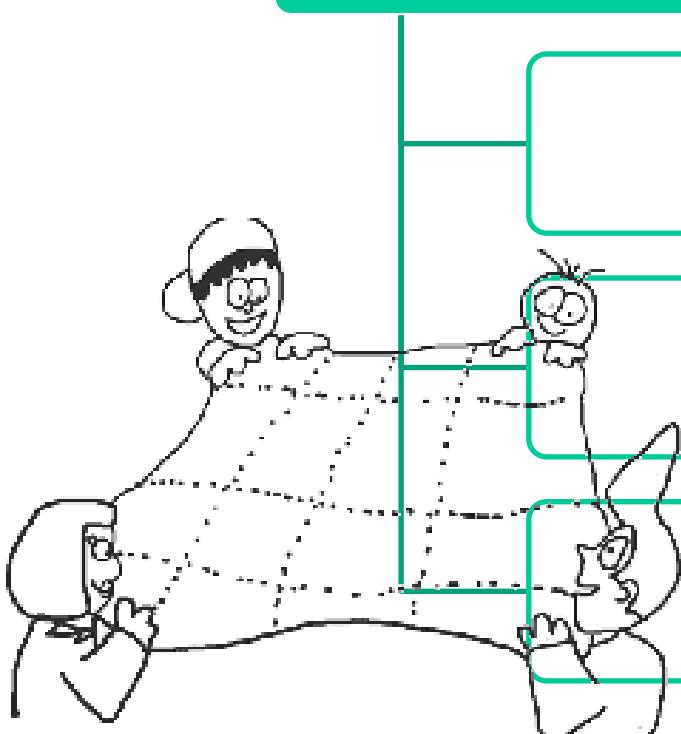
# Value of Test Design



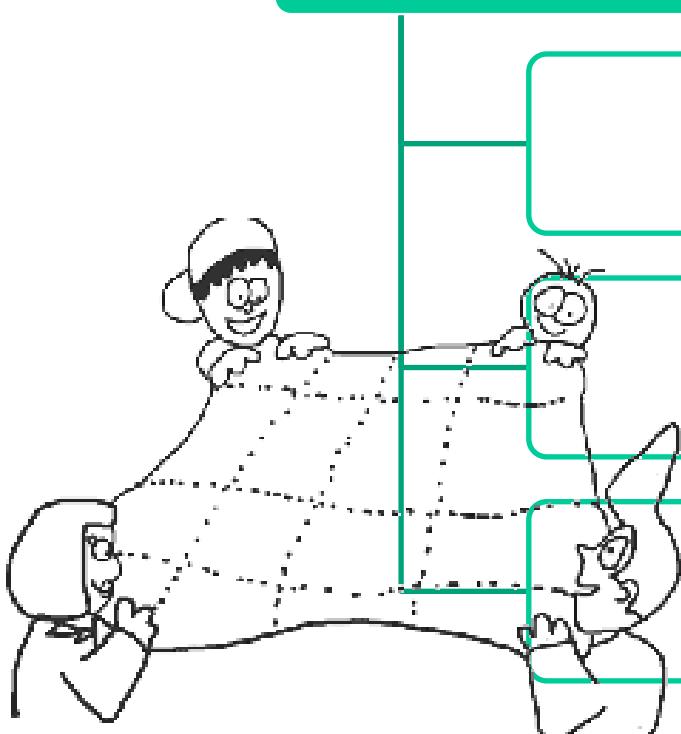
## Collaboration



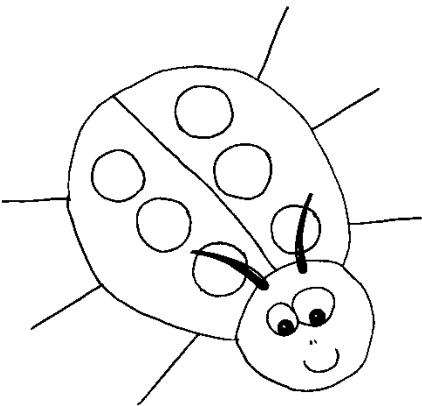
### Customers



### Developers

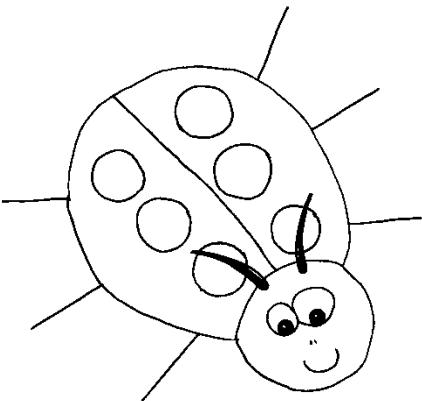


### Testers



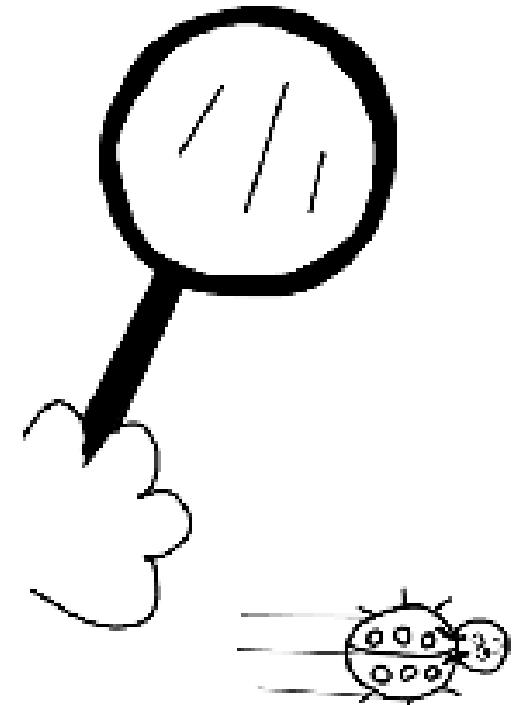
# Test Design

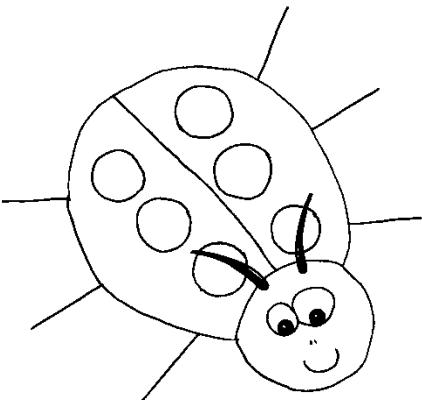
*Variable identification*



# Identify Variables

- What is a variable?
  - To VARY is to CHANGE
    - A variable is something which can change
  - Software behavior depends on the VALUES of many VARIABLES
    - Anything which influences the behavior of software could be a variable





# Identify Variables

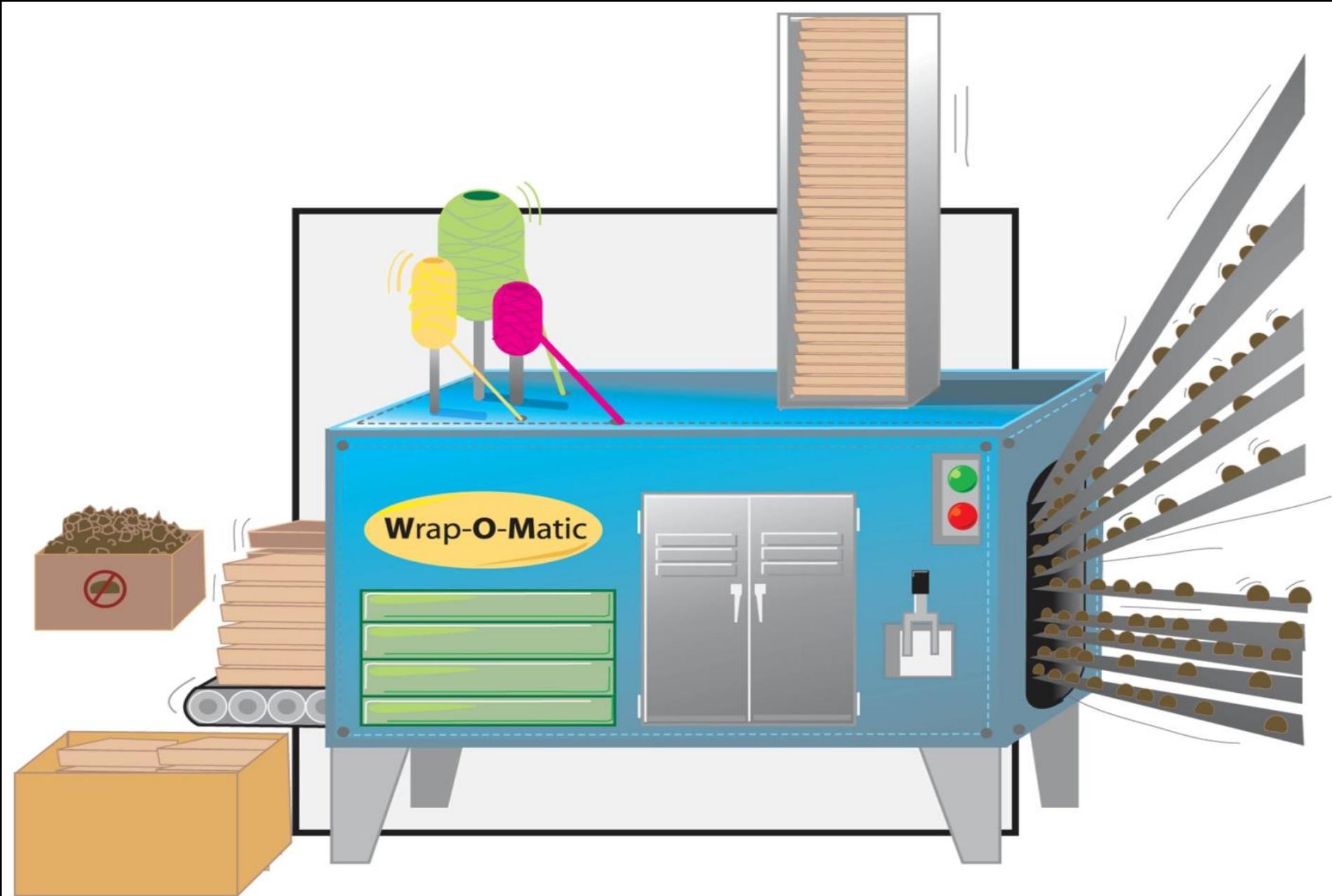
- Review test basis
- Identify variables
  - Influencers
  - Outcomes
- Potential Sources
  - Conditions
  - Environment
  - Rules
  - Constraints
  - Actions
  - States

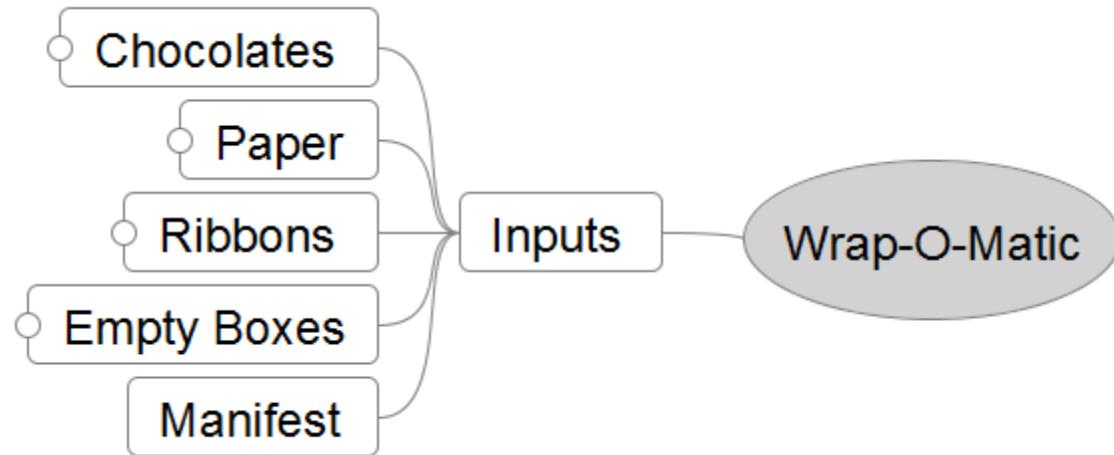


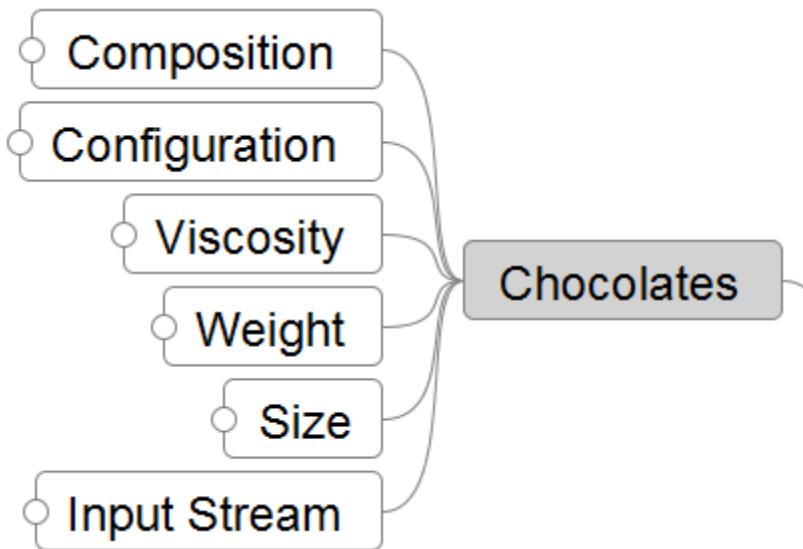


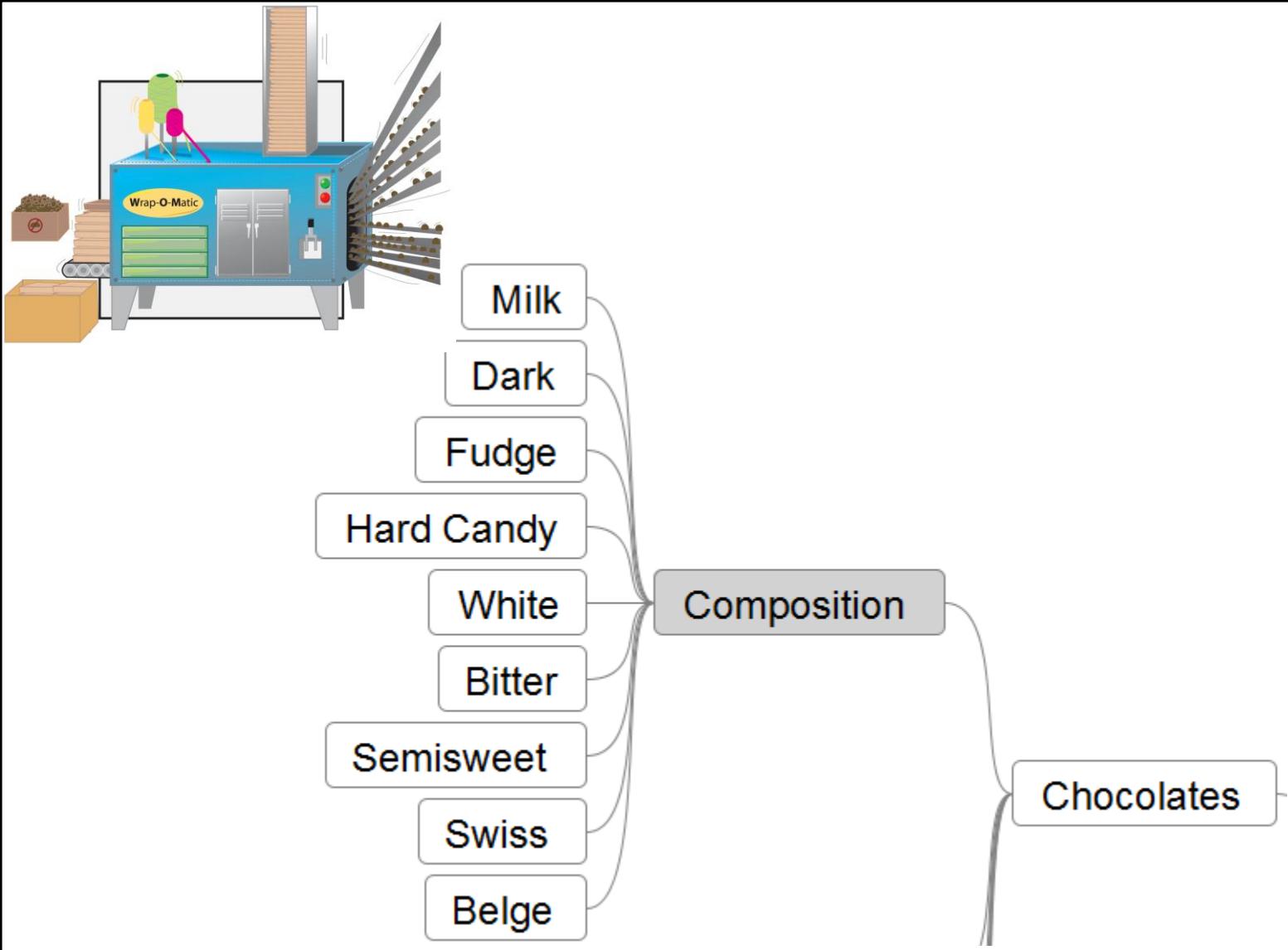
# Wrap-O-Matic

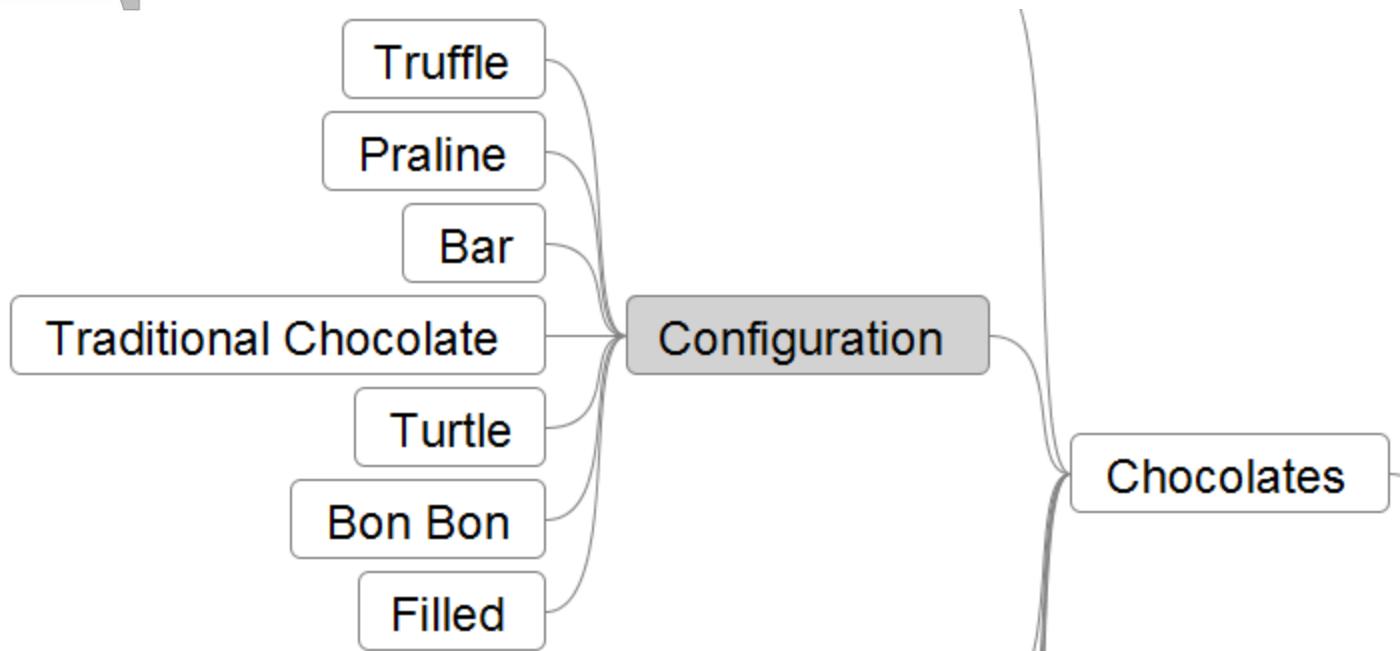
*Variable identification*

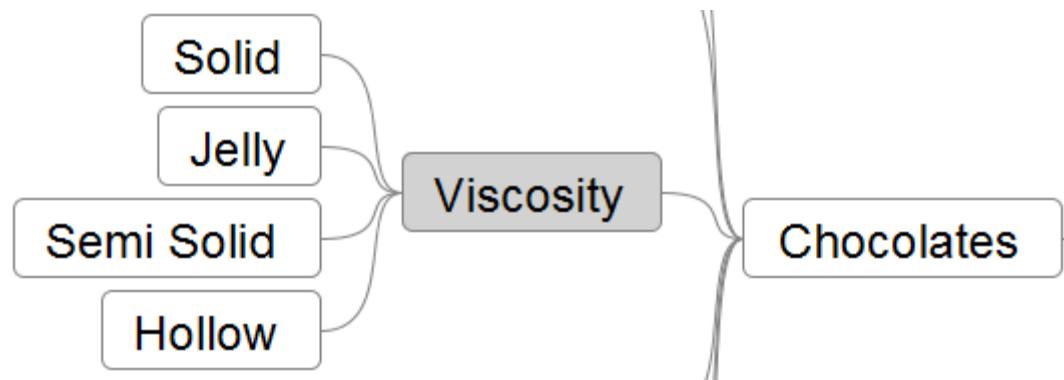


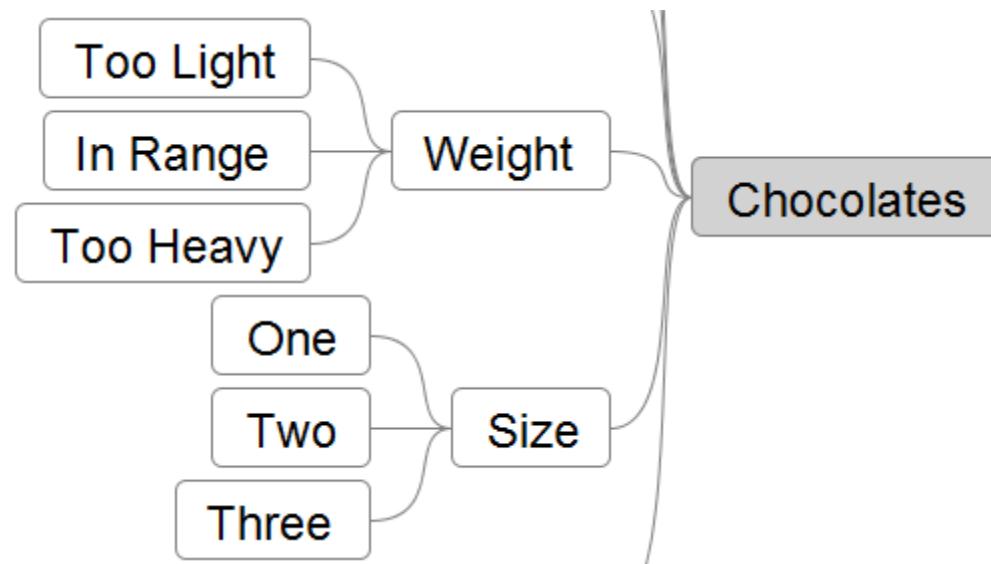


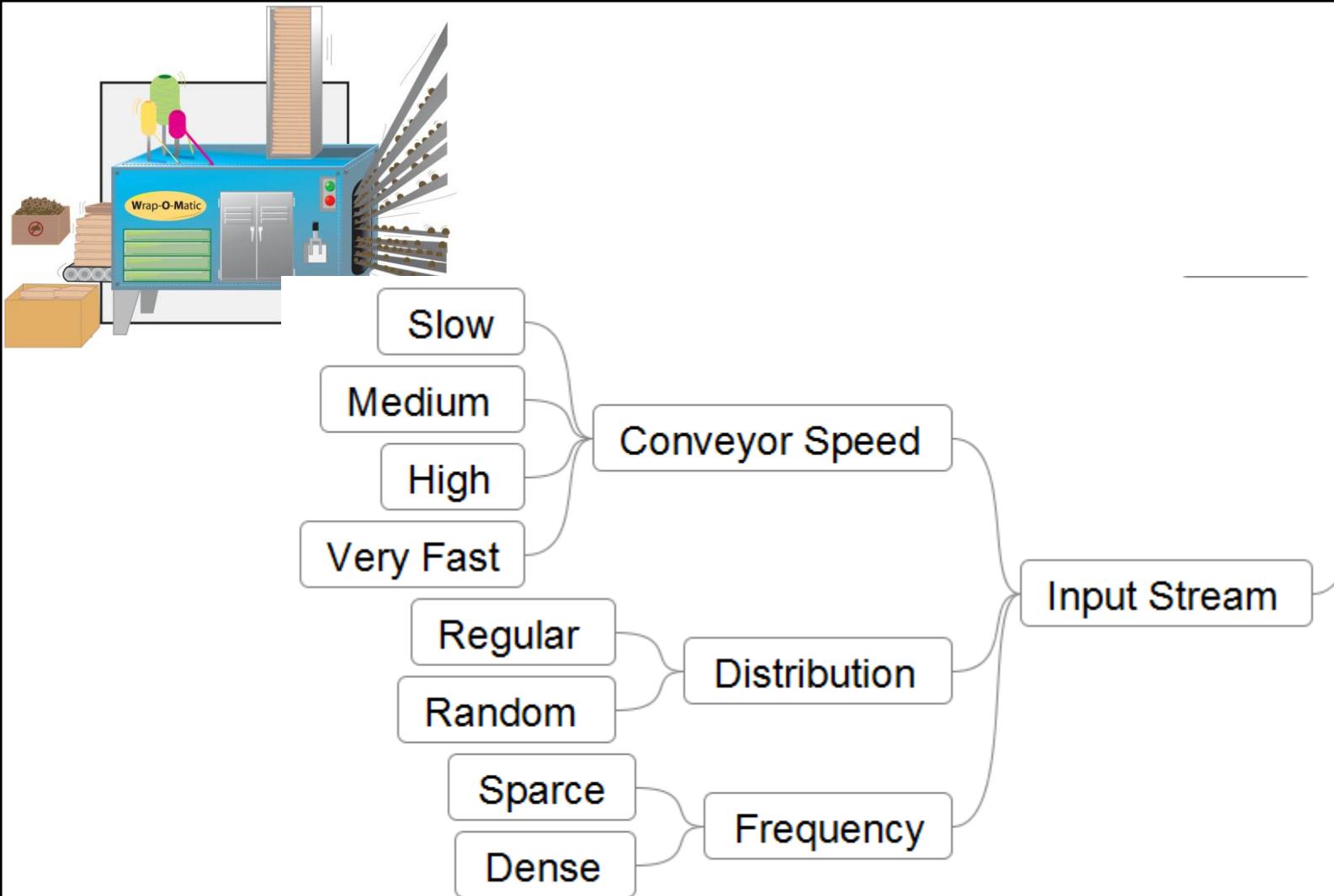


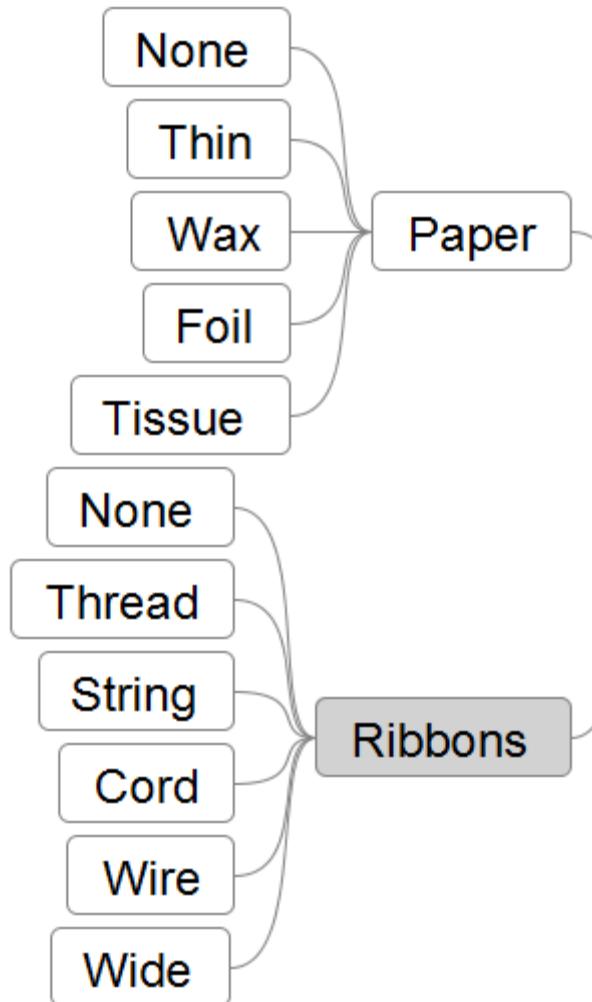


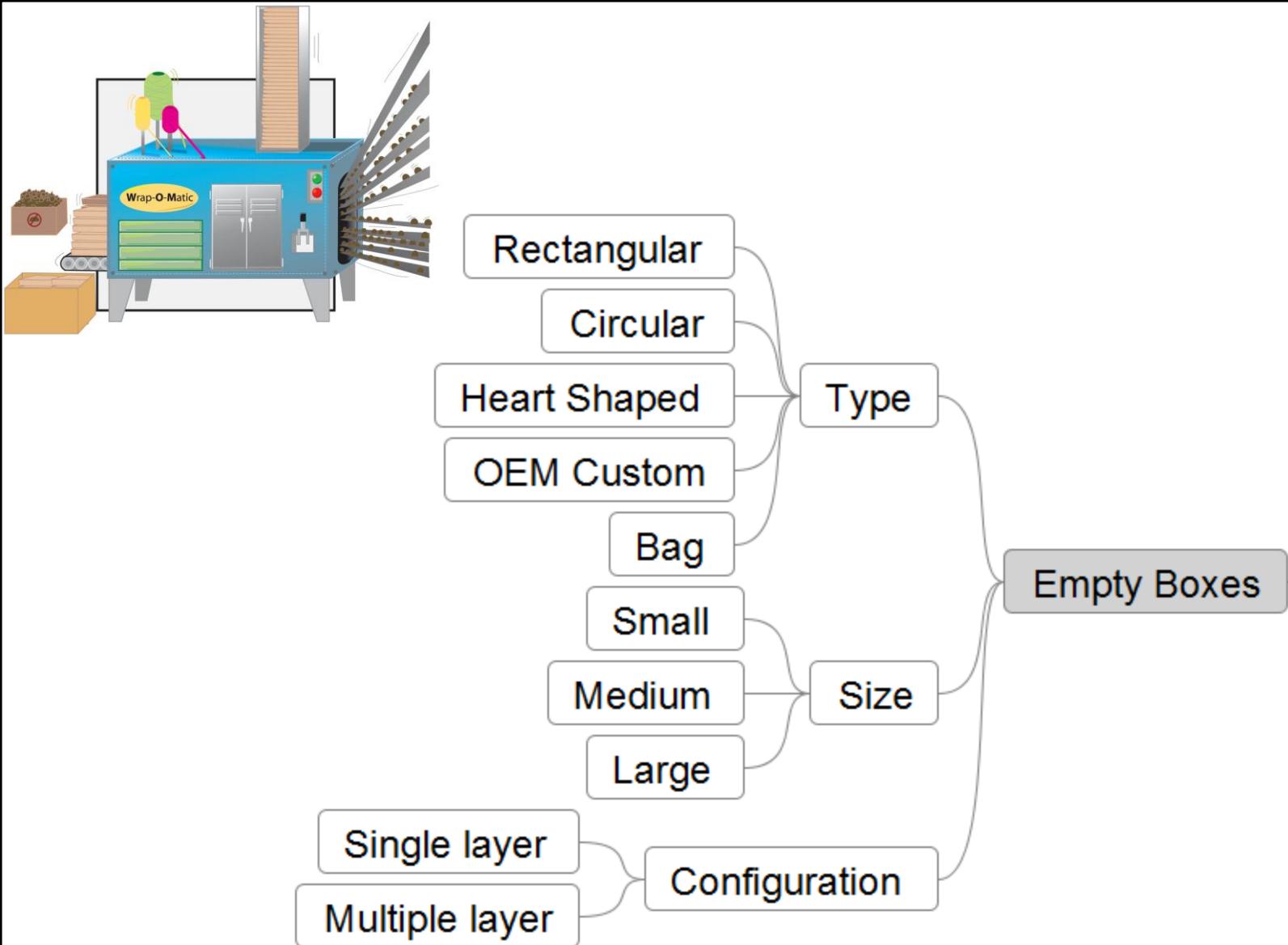


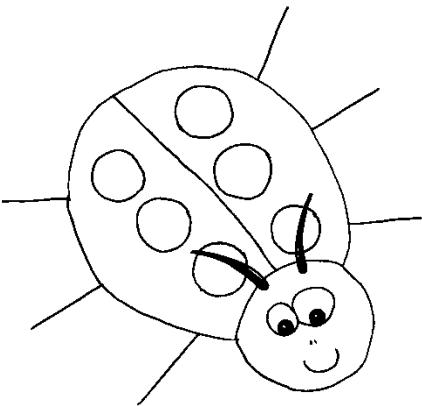






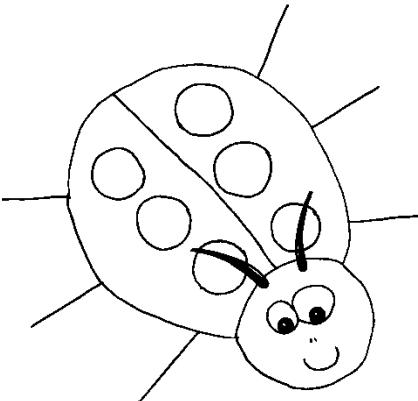






# Gmail Image Attachment

*Variable identification*



Gmail Calendar Documents Photos Groups Web more ▾

Show search results Create a Mail

Compose Mail

Inbox (746)

- Started
- Chats
- Sent Mail
- Drafts (14)
- All Mail
- Spam
- Trash

Contacts

Chat

Search, add, or invite

- Shane S. Set status here
- Salouf911 Working for the man
- Ryan Pollock
- Anthony Prusakowski
- bhutnboy@gmail.com
- Dev
- Hank Duderstadt
- honor gunday
- Illepoxy
- Ricardo Andrade
- Sara C

Date ▾ Add Contact

Labels

- trash\_archive (23)
- Edit Labels

Invite a friend

Read items from any RSS or Atom feed right here. Customize Clips

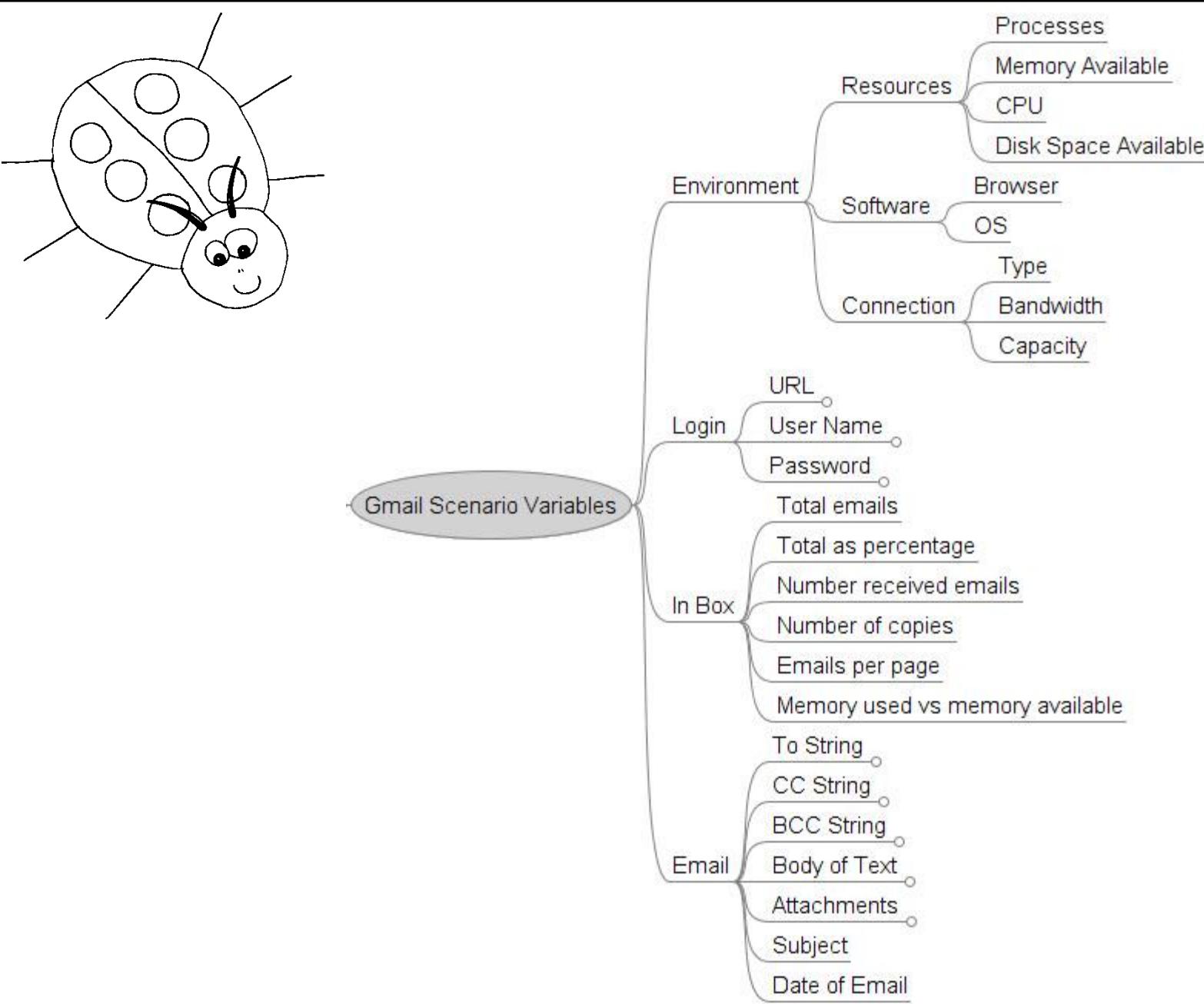
Archive Report Spam Delete More Actions Refresh

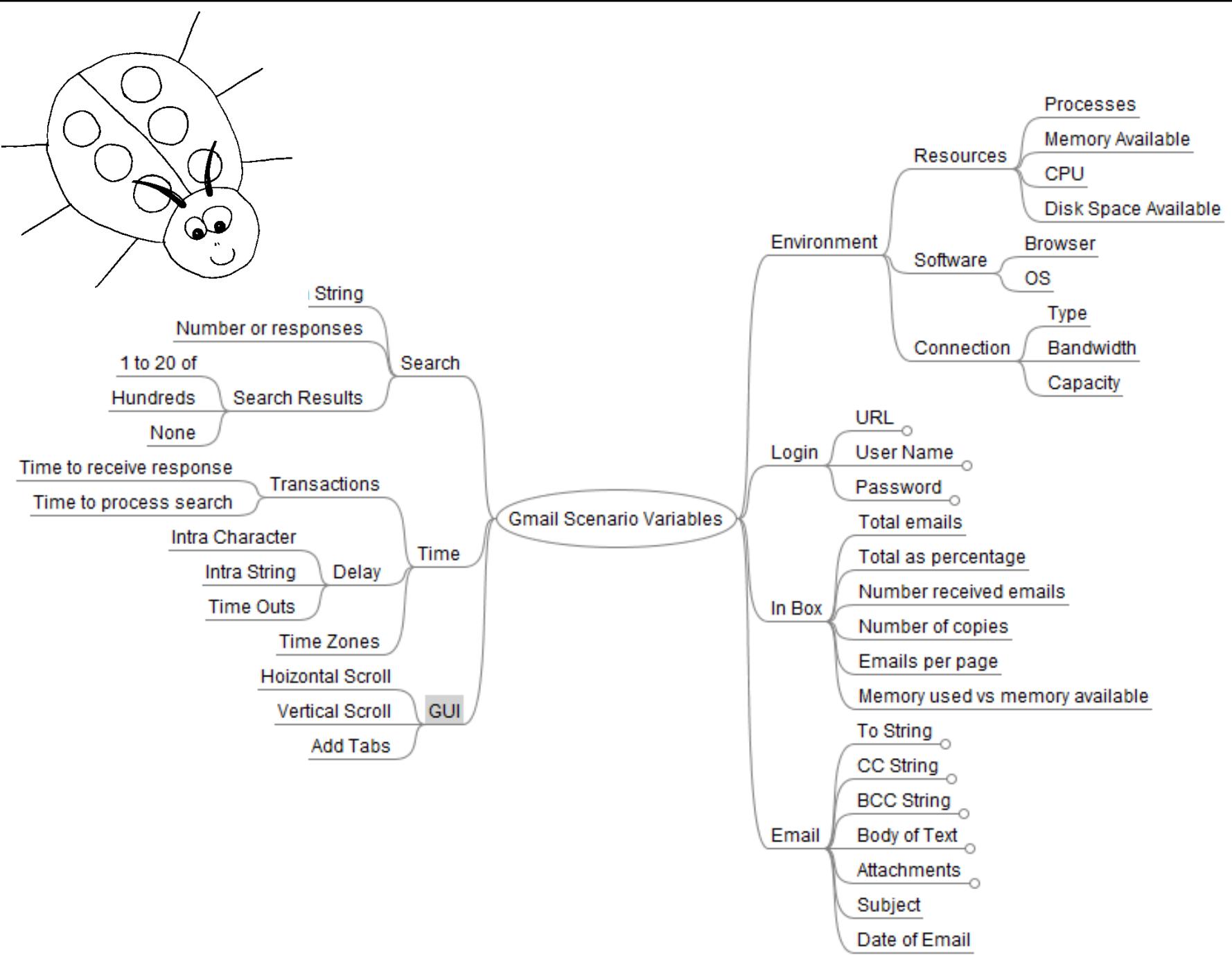
Select: All, None, Read, Unread, Starred, Unstarred

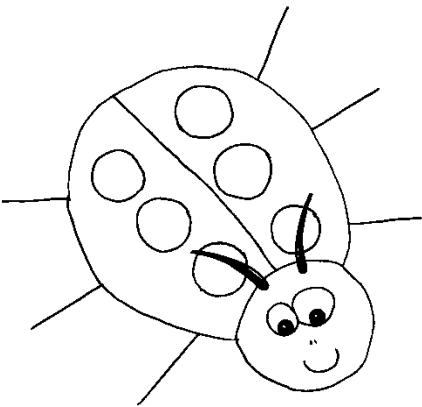
1 - 59 of 1843 Q3Mj Q3MEx

Cecanion56	christmas time - Just sharing pictures of babies... May you all have a wonderful...	Jan 16
Aparna Bajaj	you - your phone... how goes it... just wanted to say I really appreciate u helping...	Jan 10
Leslie Canoy (2)	Join my network on LinkedIn - Kristin Rivera, a Colleague at Transvideo Studio	Jan 9
Cecanion56	(no subject) - I've found a new job for me but nothing for Dad can he hold onto	Jan 9
★ me, Sharon (6)	New stuff needs your critique - Reduce the size of the h1 and its line spacing.	Jan 4
Daniel Chen	Merry Christmas and Happy New Year - Wish everyone a Merry Christmas an...	12/25/07
David Camrera	Join my network on LinkedIn - David Camrera, a Classmate at Art Institute Of...	12/25/07
Karin Andrade	Join my network on LinkedIn - Karin Andrade, a Colleague at Transvideo, inc	12/18/07
Guru.com	Welcome to Guru.com - Guru.com   Welcome to Guru.com	12/18/07
The Art Institutes Alumni	The Art Institutes Alumni News Letter - This message is being sent to you because...	12/17/07
Denis Demenbegovic	Resume - henris resume Be a better friend, neighbor, and know-it-all with You...	12/16/07
★ Elexis, me (4)	(no subject) - http://users.livejournal.com/_publicuse/248885/item?mode=reply	12/14/07
alex park (2)	Transvideo Christmas Party - FYI - \$10.00 is a suggested amount. Please give...	12/10/07
Dechen Wangdi	Check out my Facebook profile - Facebook Dechen Wangdi Dechen has 675 fri...	12/10/07
Cecanion56	Fed: PWK Got my tree up at last night... - Check out AOL Money & Finance Ju...	12/6/07
randotrade@gmail.com	TV Corporate Website Copy - I've shared a document with you called "TV Cor...	12/3/07
Rebecca Petter	Fall 2007 Portfolio Show - Hey Shane, This is Bailey, I don't know if you remem...	12/2/07
Dessart, Donna	The Hottest Spot on the Line! - Dear AlCASS Alumni: We continually invite you to...	11/28/07
des and dot	Eve: Reminder for A Surprise Birthday Party for Eugenio! - evie Hi alclass...	11/22/07
Sharon Keltner	Fed: COGI needs people ... talented and preferably demented people ... A re...	11/19/07
Southwest Airlines	Ticketless Travel Passenger Itinerary - Southwest Airlines Travel Itinerary Thi...	11/18/07
LinkedIn Updates	LinkedIn Network Updates, 11/17/2007 - LinkedIn NETWORK UPDATES Post	11/17/07
me, Elexis (3)	Hi - holy crap. Greyhound's ridiculous. It's gonna cost me almost \$40 to go to L...	11/15/07
A-Jay Jimenez (5)	Come to my next event: Fraggle Rock! This Friday, Nov. 16 2007! - Biscoe the...	11/12/07
me, Sara (2)	Hi sara chun - Cocoon Passover, On Nov 8, 2007 11:00 AM, Shane S. <shane...	11/11/07
Cecanion56, me (4)	ehas - i don't know. I think it'll be pretty miserable either way. On Nov 9, 2007 1...	11/9/07
Elexis, me (7)	(no subject) - you always let me win, that's no fun. On Nov 9, 2007 2:00 PM, E...	11/8/07



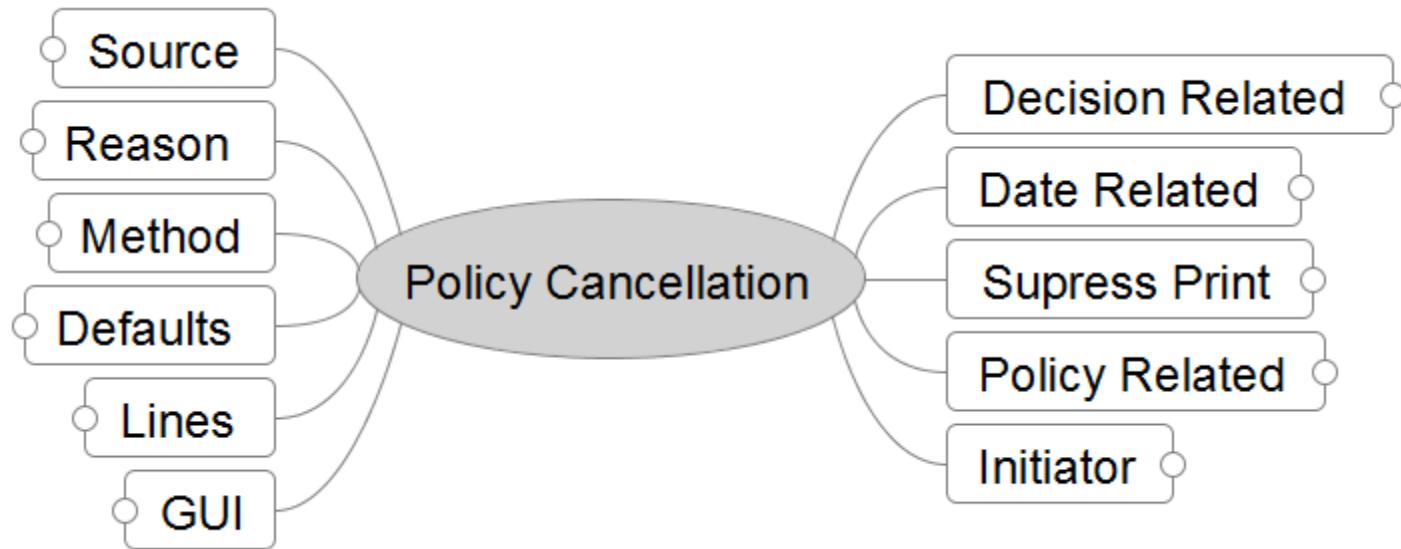
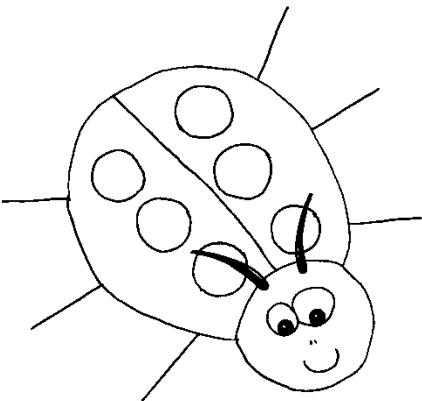


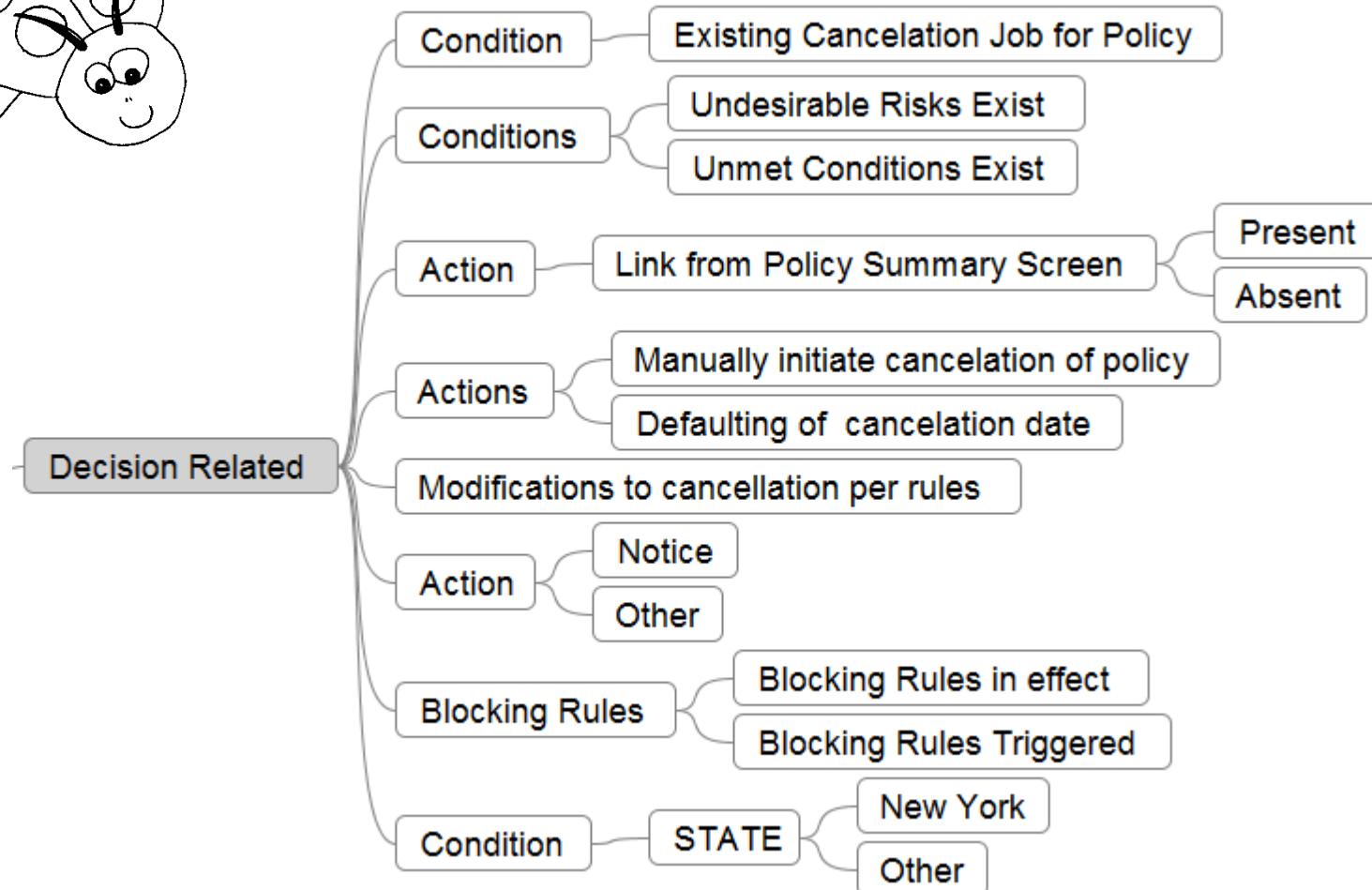
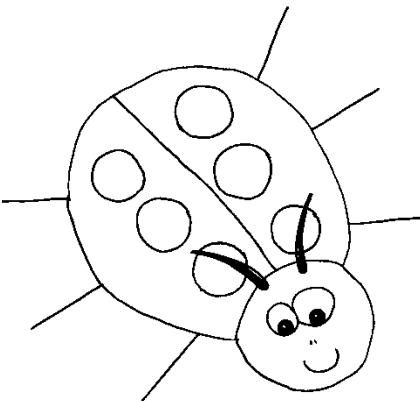


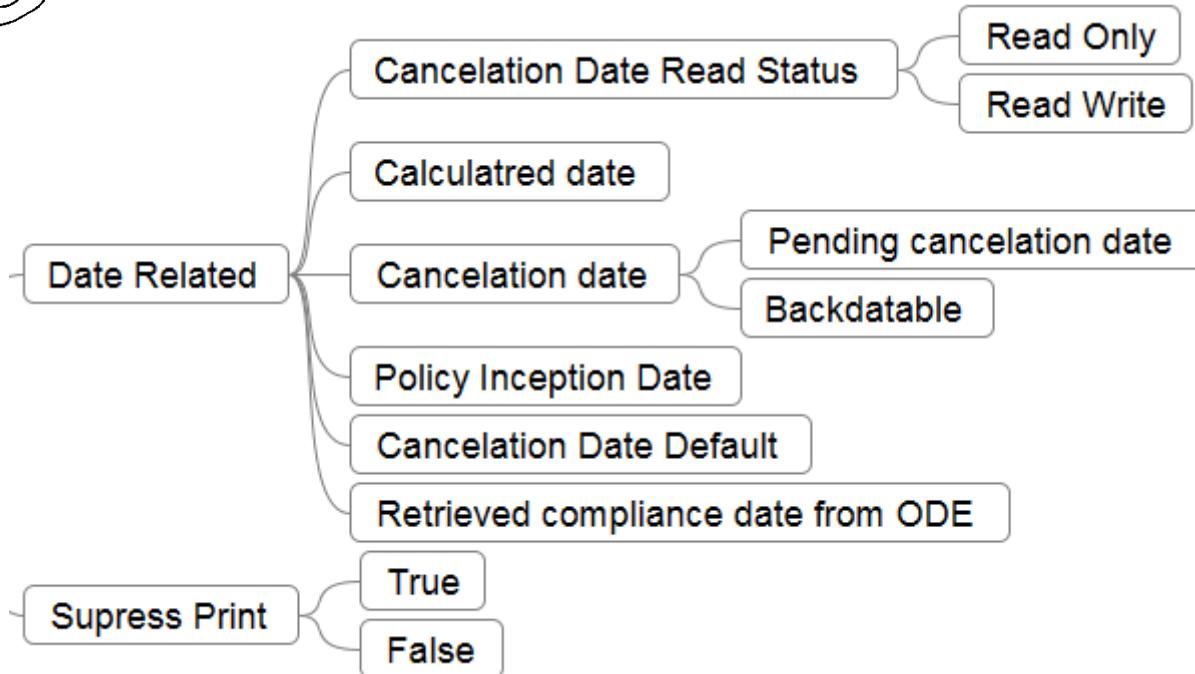
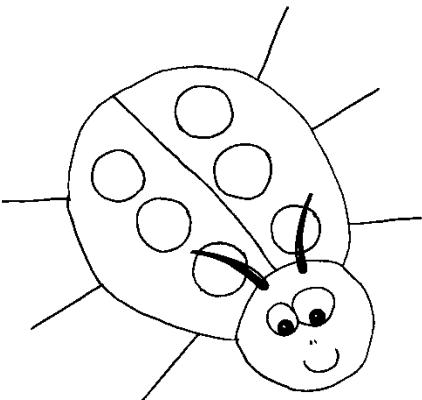


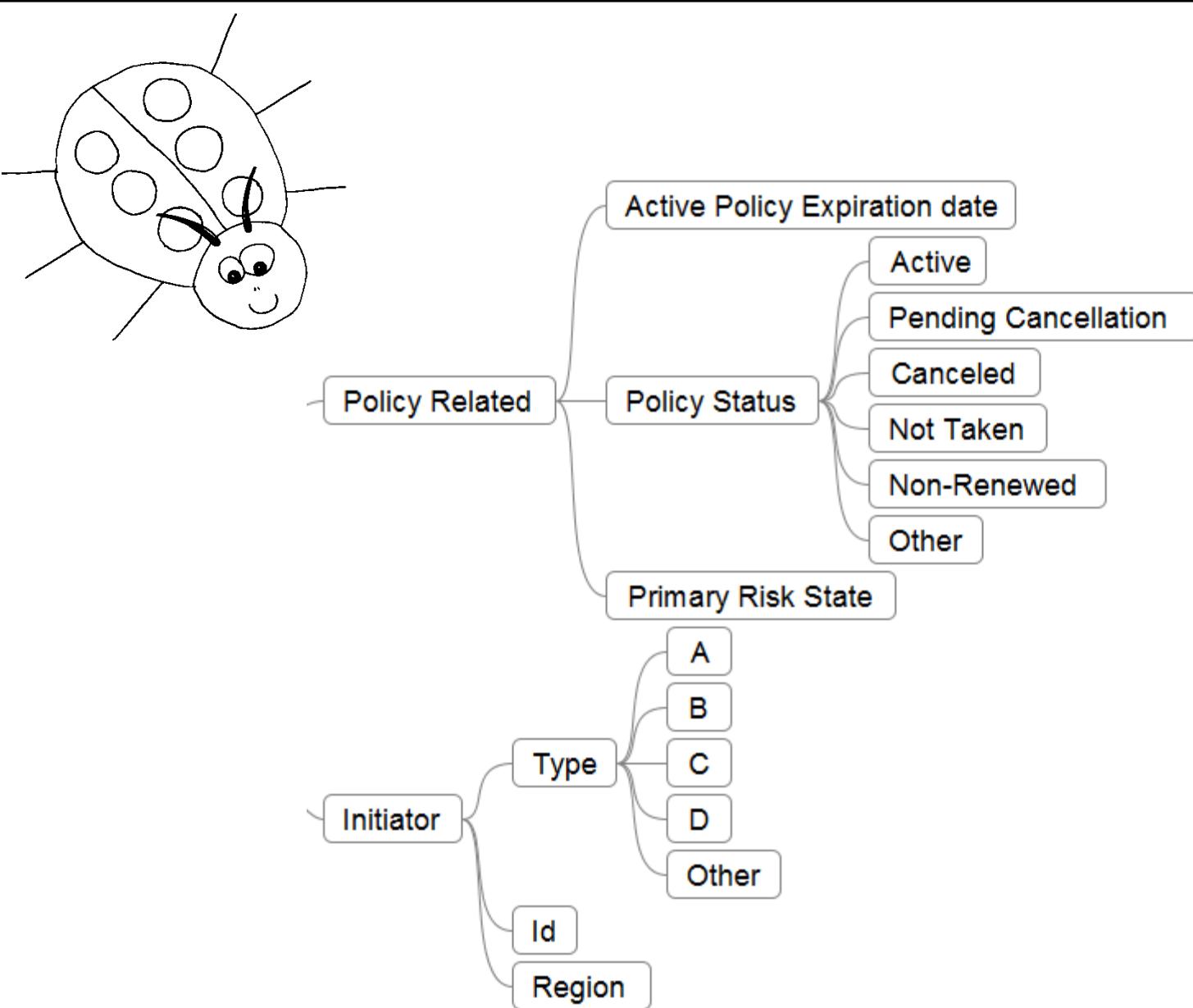
# Insurance Policy Cancellation

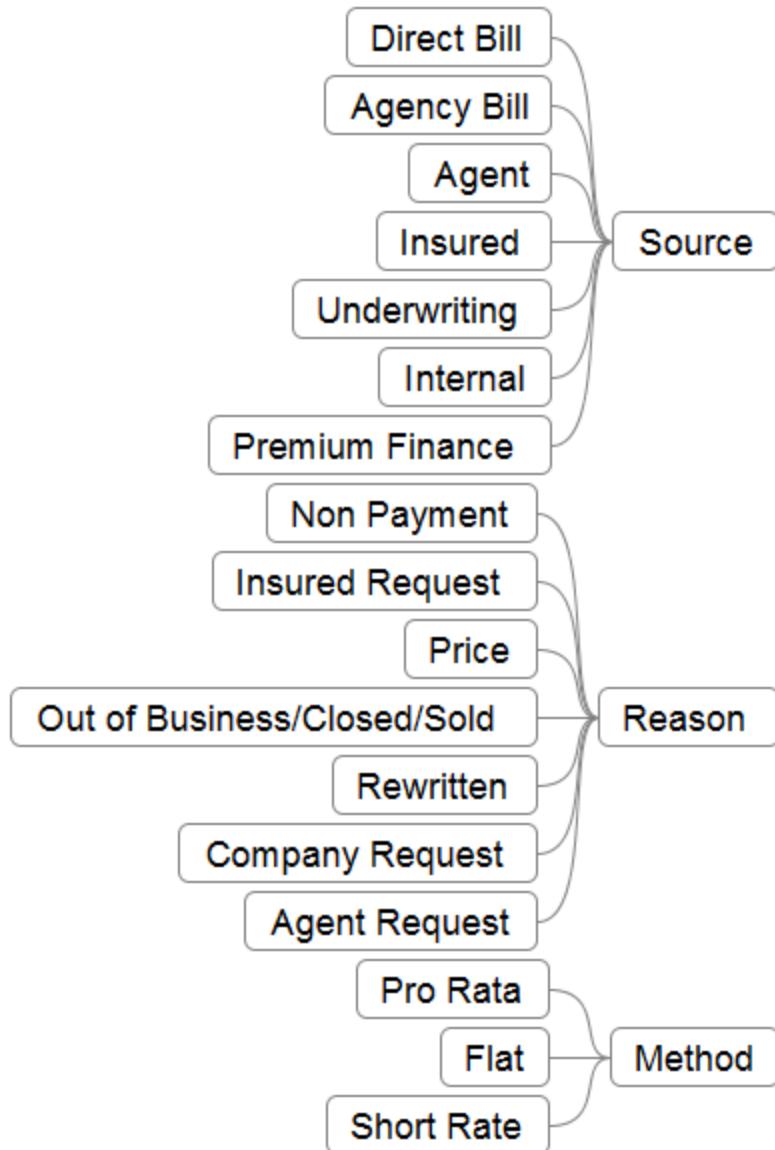
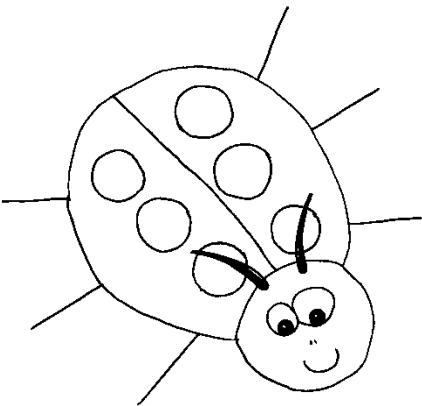
*Variable identification*

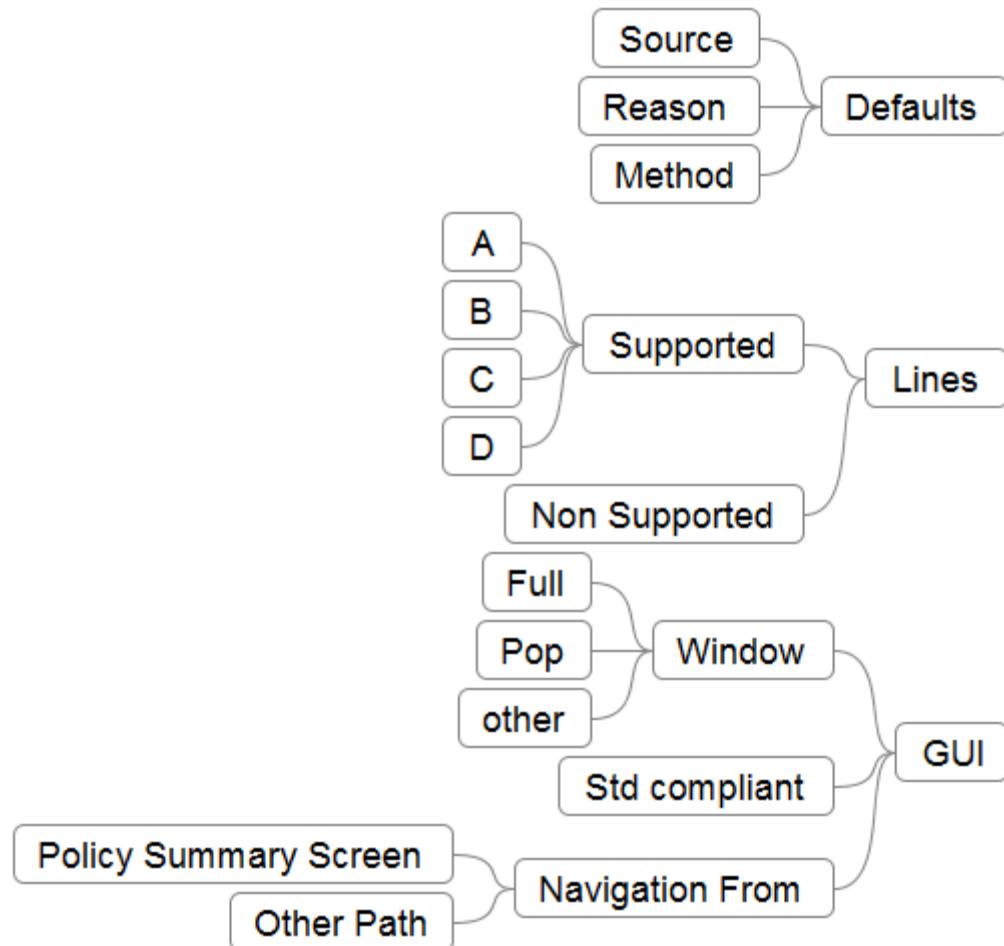
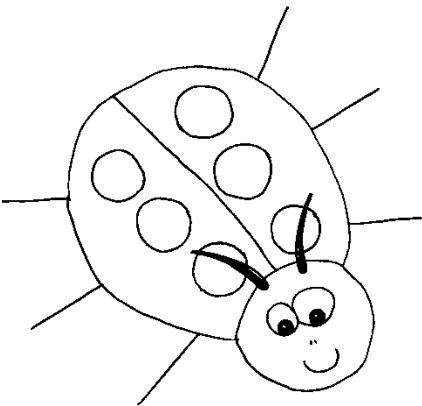


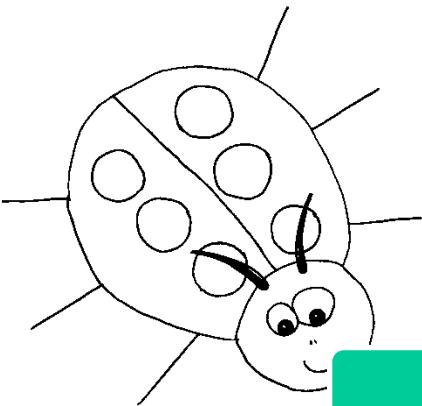












# Identify Variables

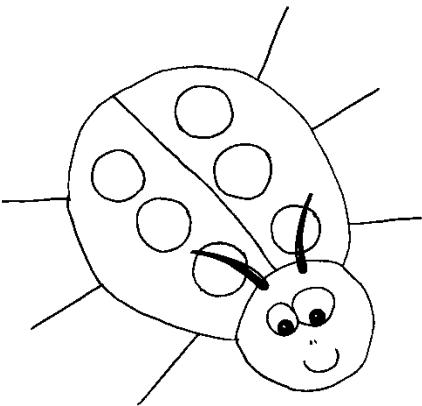
Act on variables

Ignore

Default values

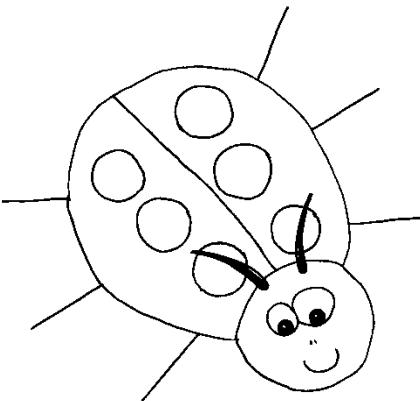
Specific values

Observe



# Test Design

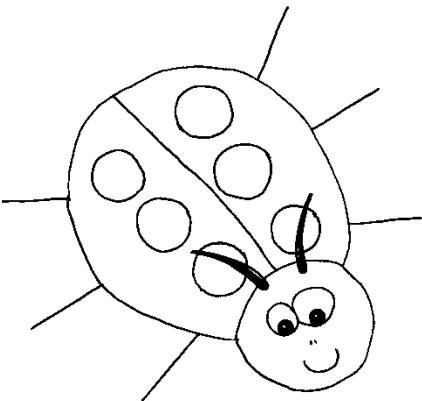
*Equivalence Classes*



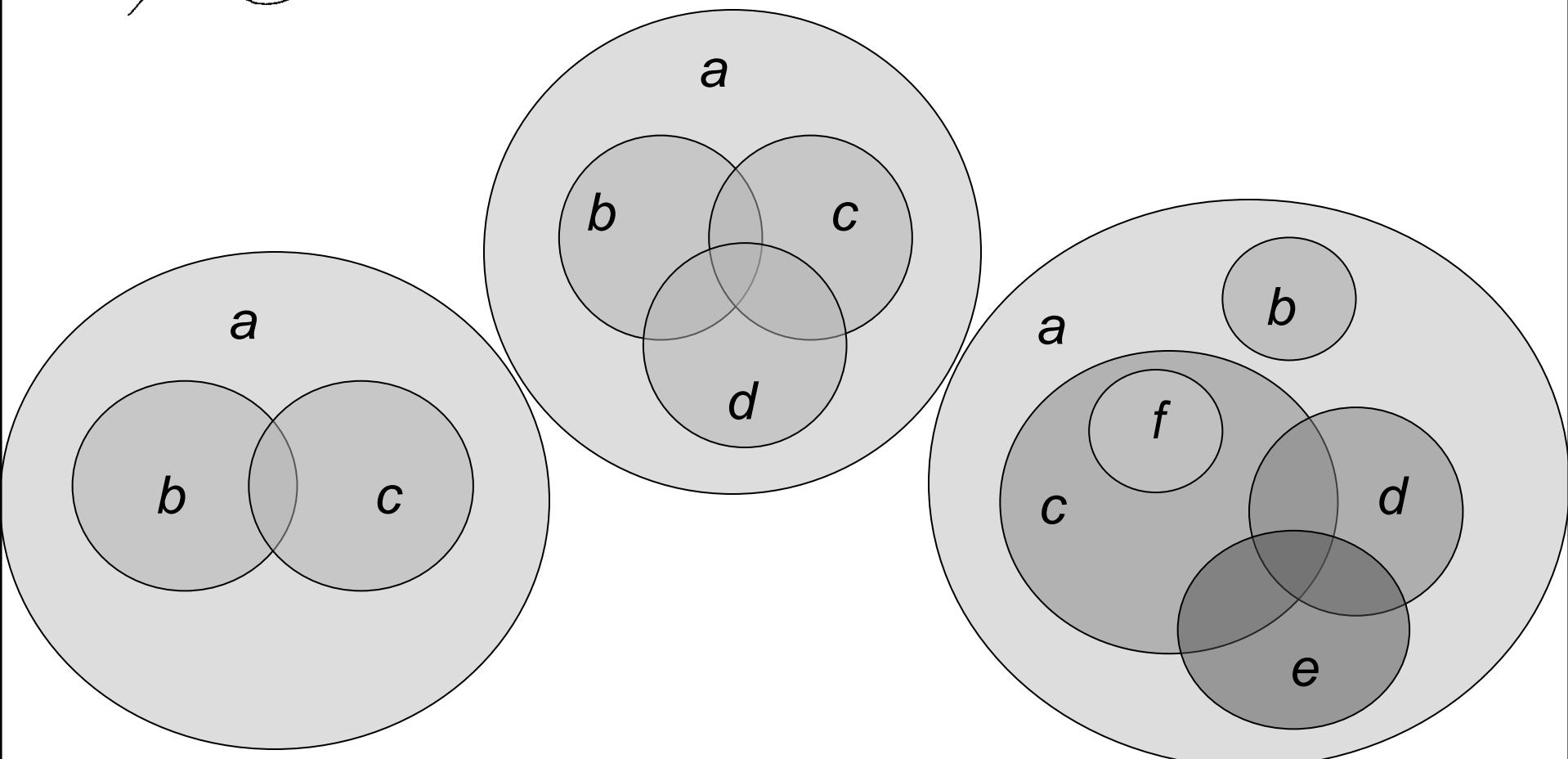
# Taking AIM

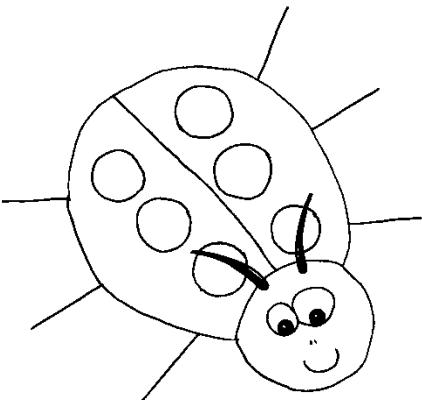
- Equivalence class
  - A *subset* of all possible test values to a variable
  - Each member assumed provide the same info
  - Each variable may have *many classes*
  - Equivalence class are *not mutually exclusive*
  - Focus testing
  - Reduce the number of test cases





# Sets - Venn Diagram Equivalence Classes

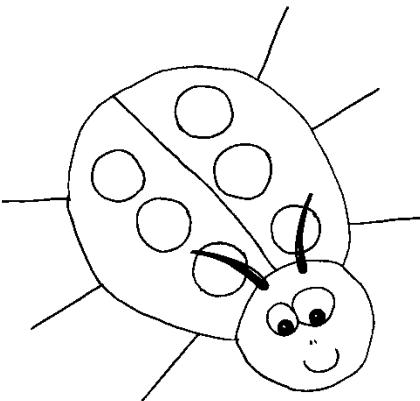




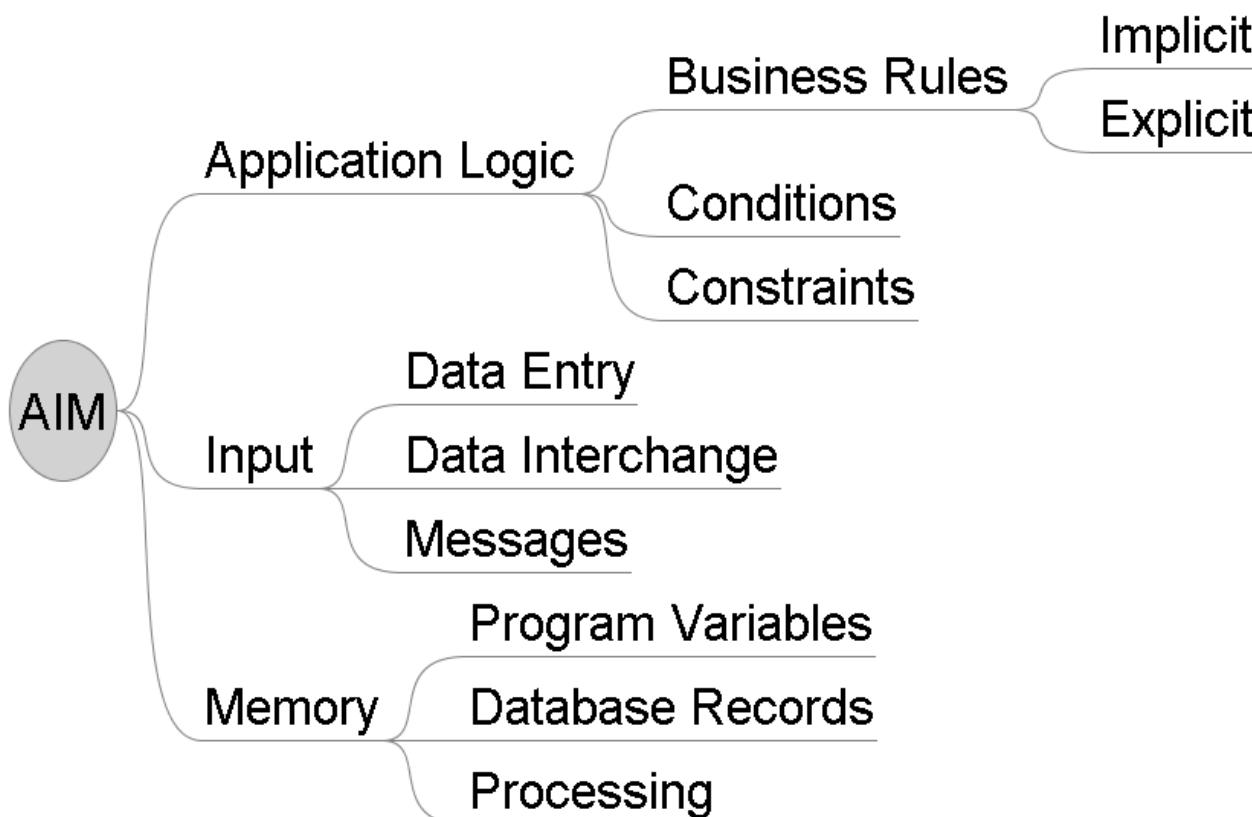
# Equivalence Classes

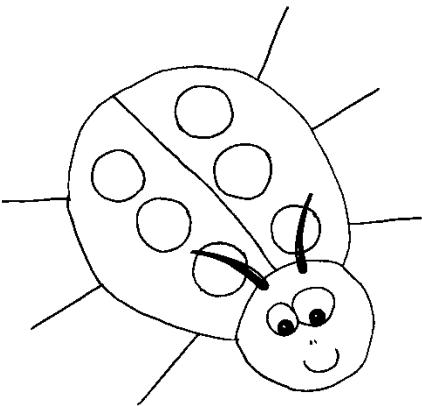
- Sources
  - Requirements
    - Business logic
    - Capabilities
    - Ranges
    - Constraints
  - Code
    - Decisions
    - Intermediate computations
  - Data
    - Input fields
    - Database
    - Internal structures





# AIM - Equivalence Classes





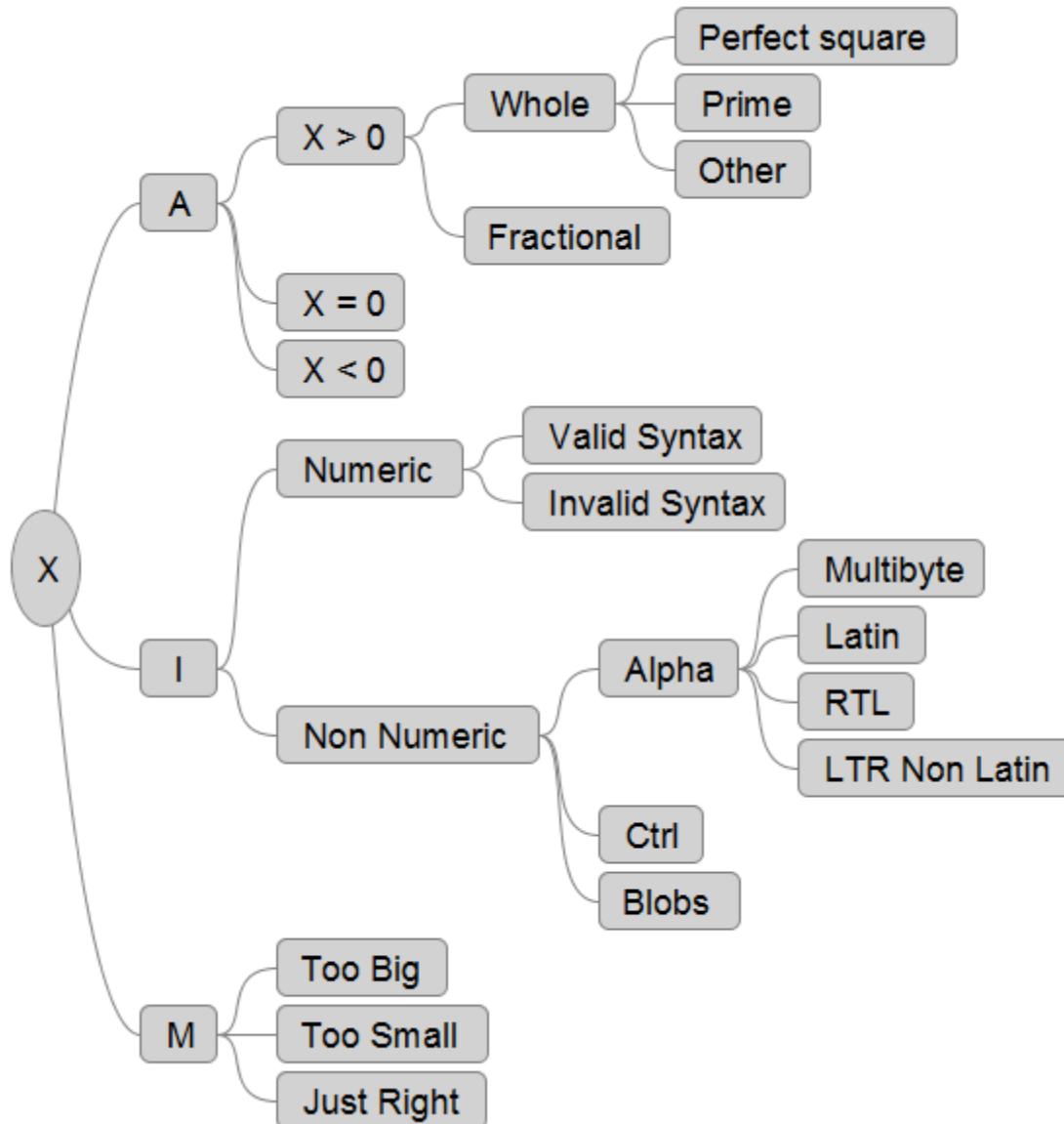
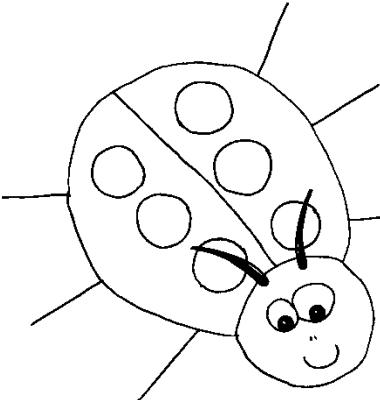
# Square Root Function

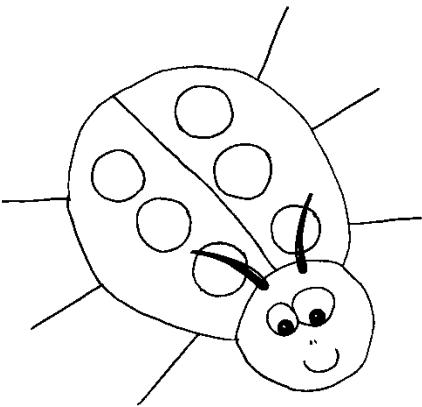
*AIM - Equivalence Classes*

# Example: SQRT X

## Equivalence Classes

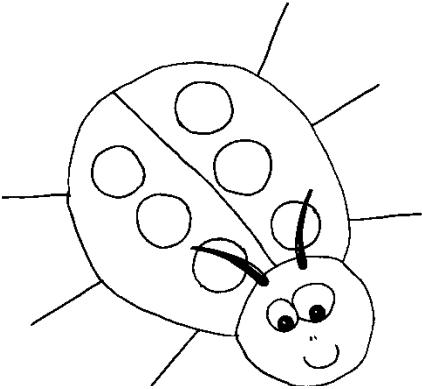
)



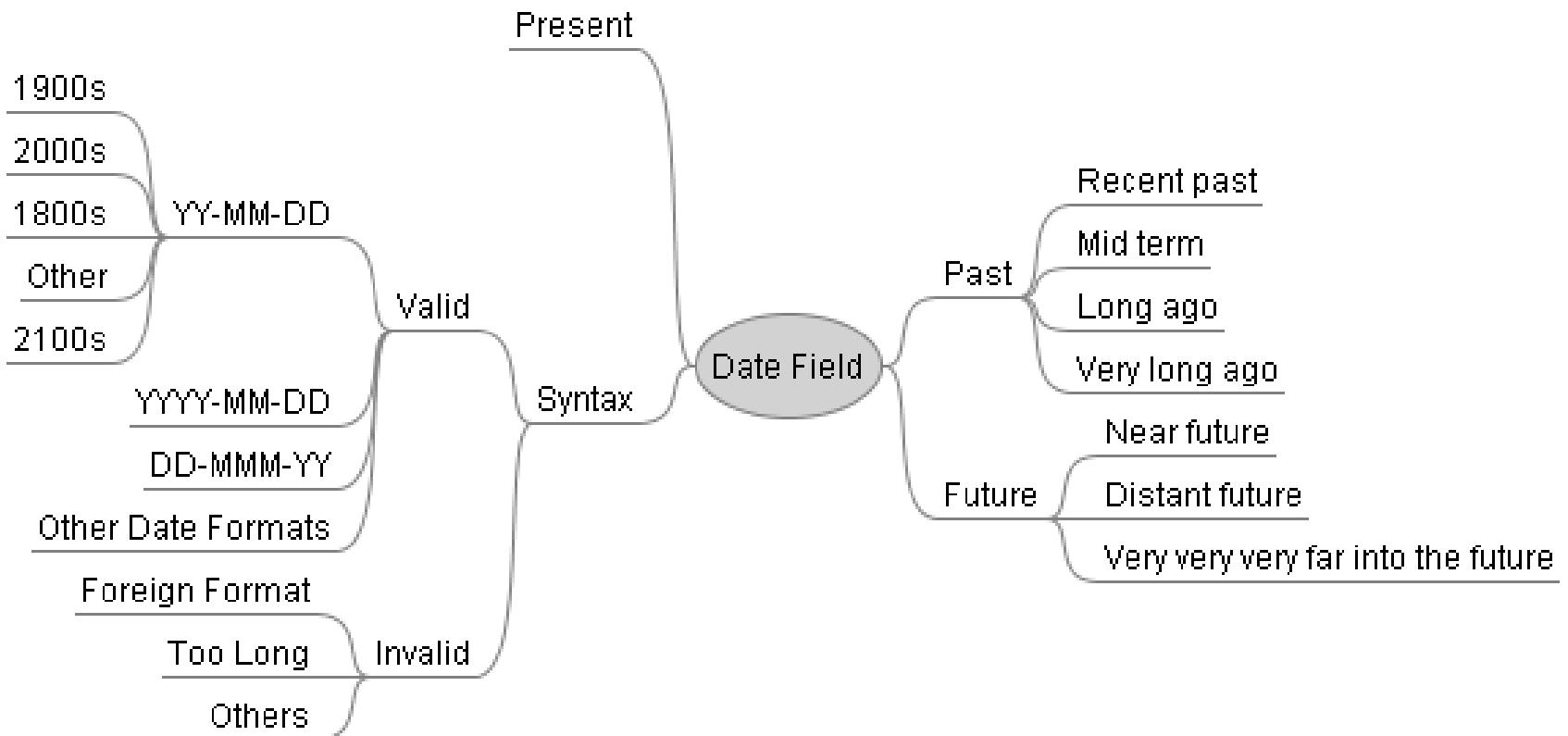


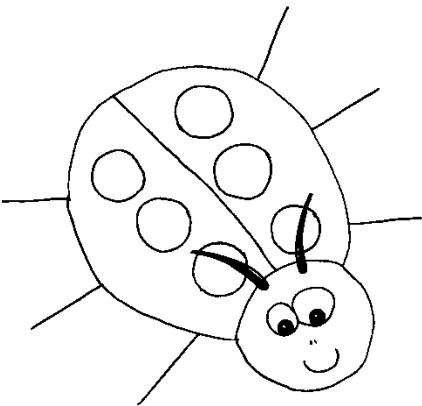
# Date Field

*AIM - Equivalence Classes*



# Example: Date Field Equivalence Classes Mind Map

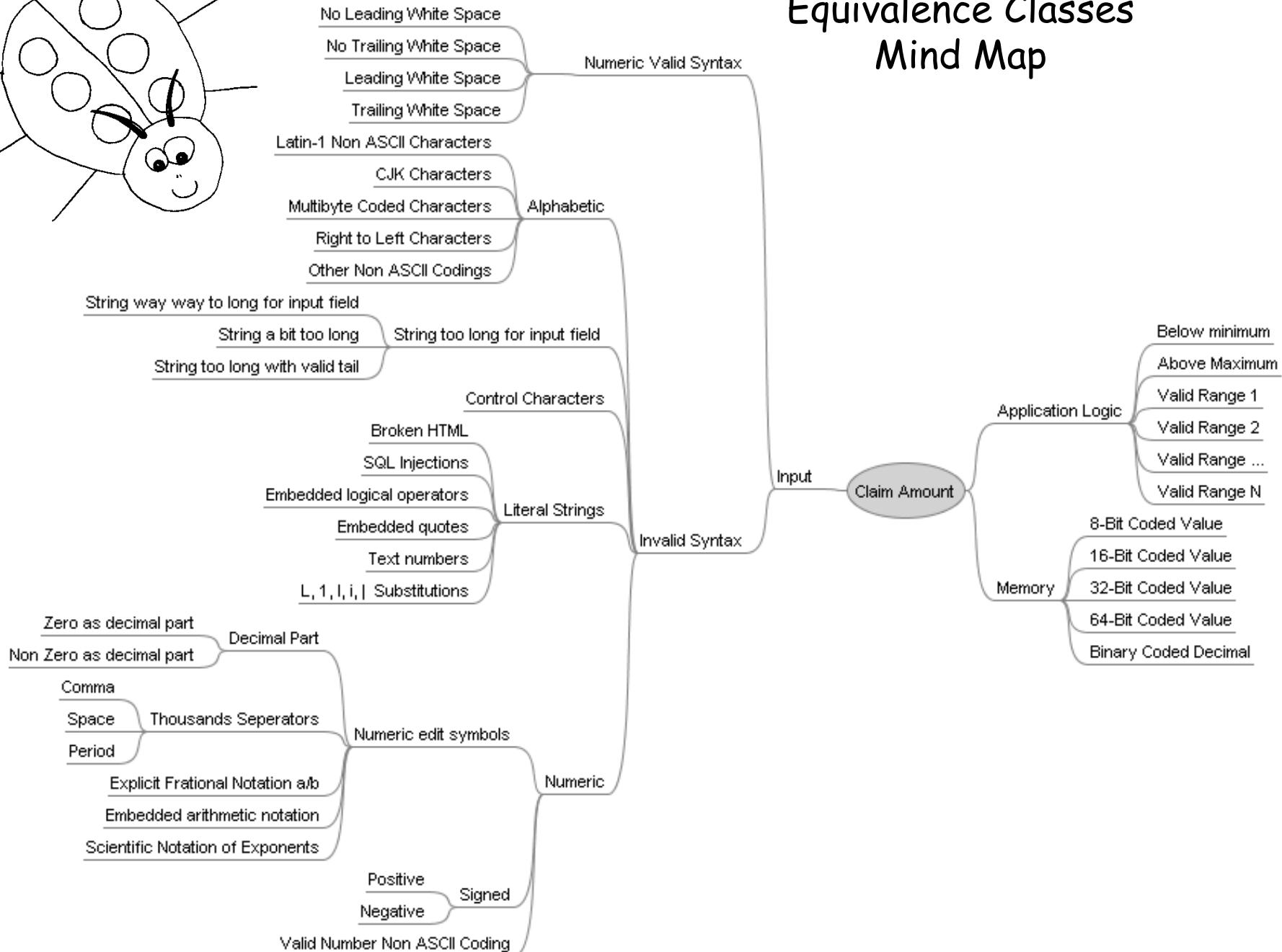
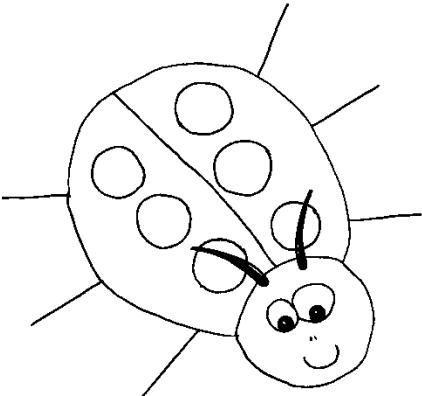


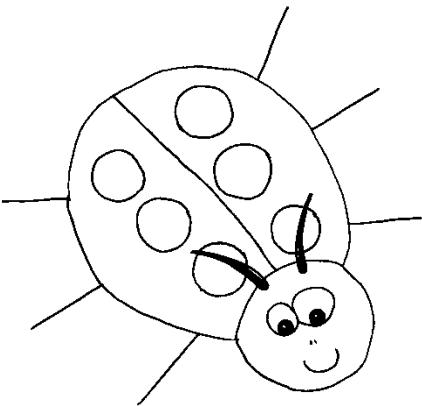


# Insurance Claim Amount

*AIM - Equivalence Classes*

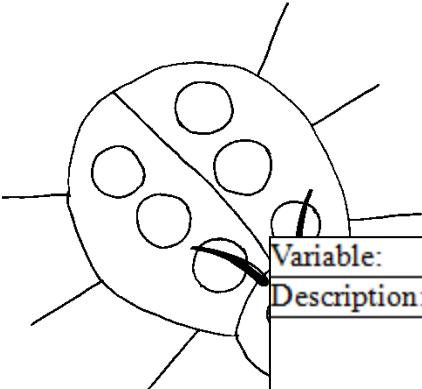
# Example: Claim Amount Equivalence Classes Mind Map





# Gmail “to composed”

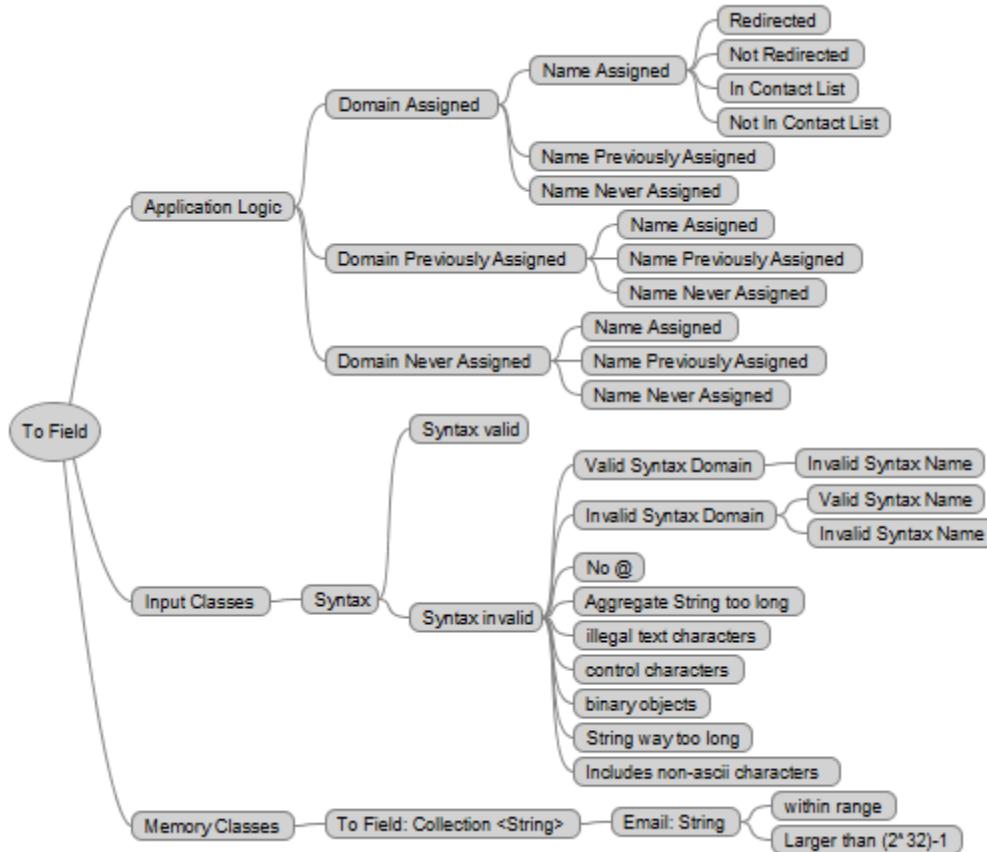
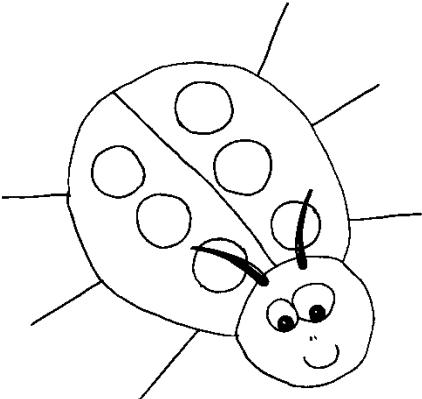
*AIM - Equivalence Classes*

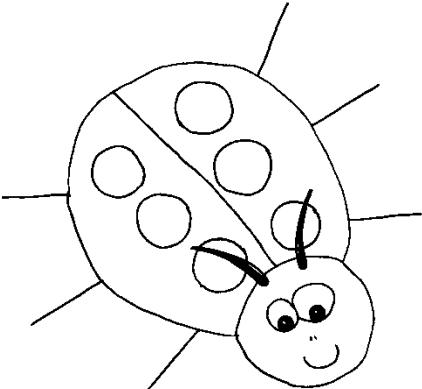


# Example: Gmail To Composed Equivalence Classes Mind Map

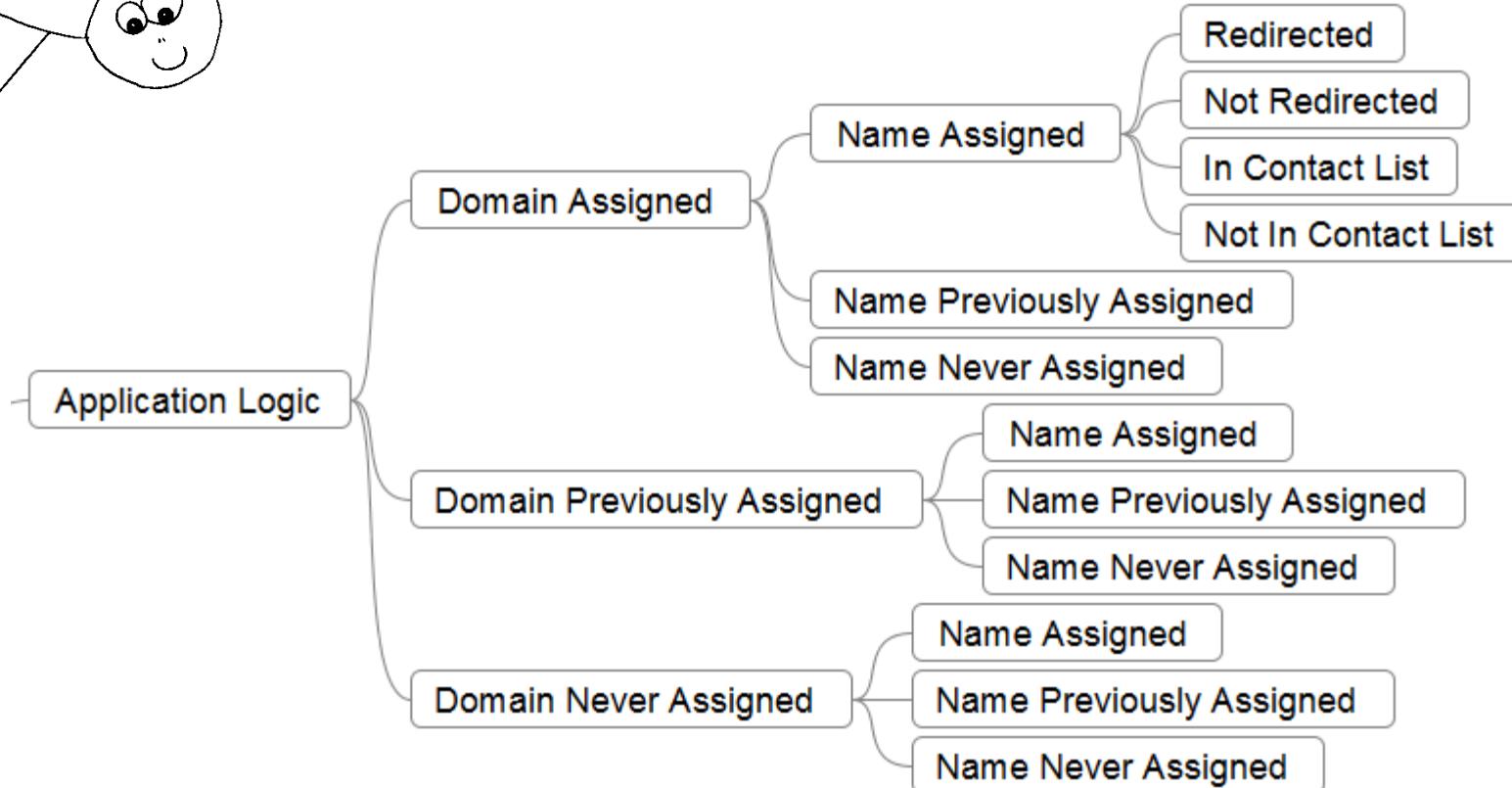
Variable:	To (composed email)
Description:	This is the To field in a composed email
	<b>To:</b> receipt@mail.com  <b>Cc:</b> _____  <b>Bcc:</b> _____
Gmail API Source:	g4j-gmail_api/GMComposedMessage.java (Line 19)
Other Source:	Java Collection: <a href="http://java.sun.com/j2se/1.4.2/docs/api/java/util/Collection.html">http://java.sun.com/j2se/1.4.2/docs/api/java/util/Collection.html</a> A collection object is used to implement a list to store emails  Java String Implementation: <a href="http://www.docjar.com/html/api/java/lang/String.java.html">http://www.docjar.com/html/api/java/lang/String.java.html</a> (Line 119: number of characters is stored as an <i>int</i> )  Java <i>int</i> Implementation: <a href="http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Integer.html">http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Integer.html</a> (constant MAX_VALUE states that maximum value of int is $2^{31}-1$ , thus maximum length of <i>String</i> is $2^{31}-1$ )

# Example: Gmail To Composed Equivalence Classes Mind Map

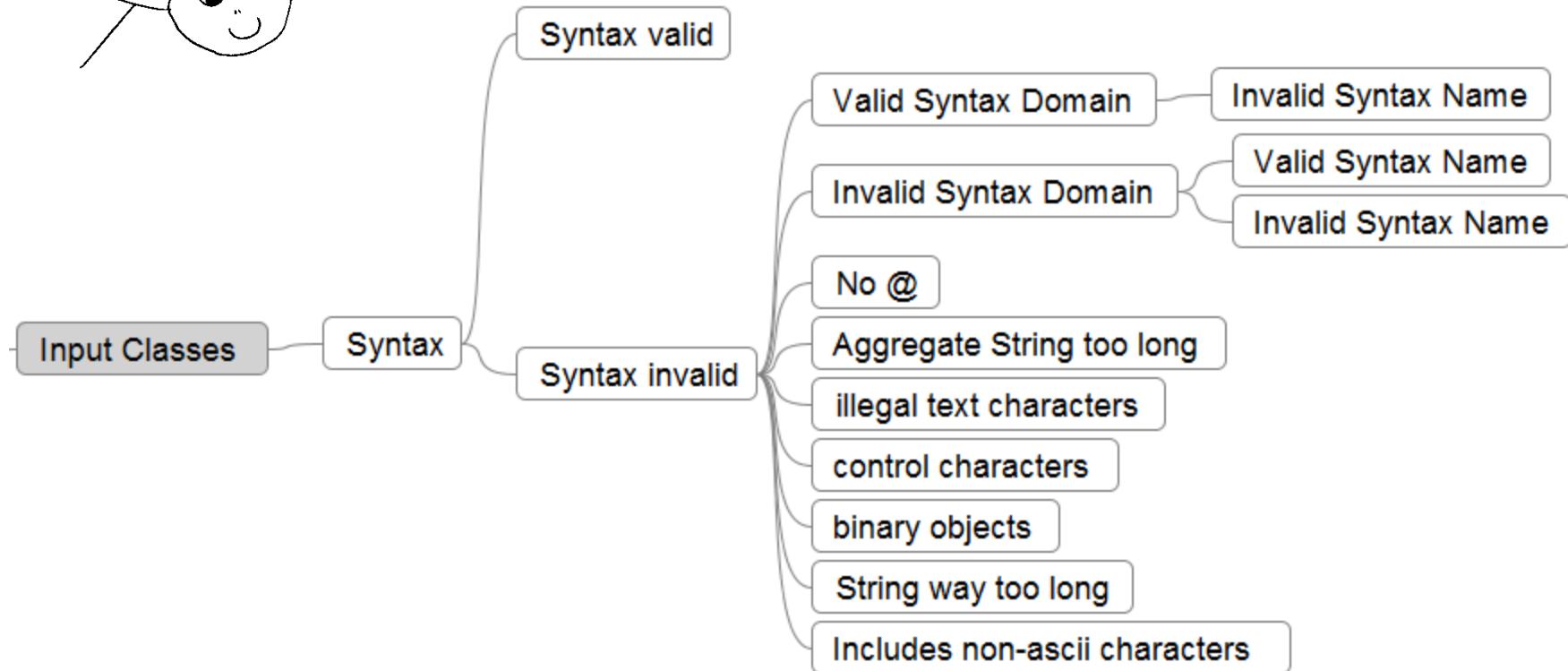
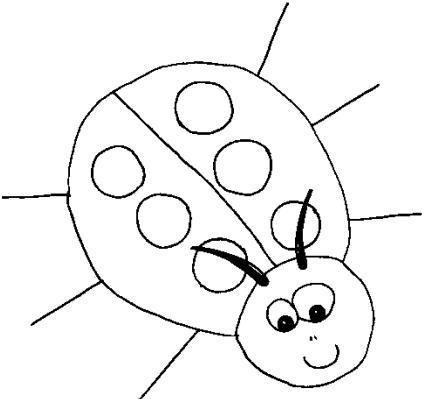


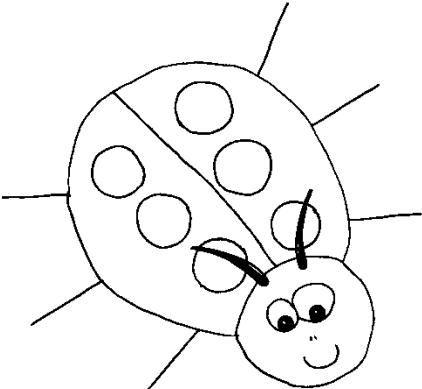


# Example: Gmail To Composed Equivalence Classes Mind Map



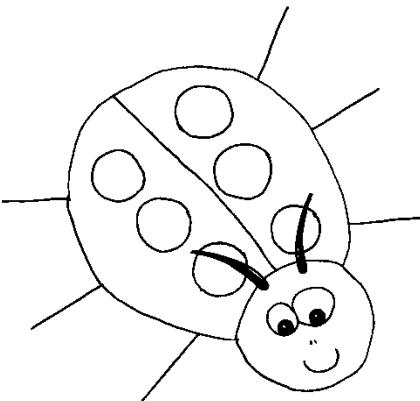
# Example: Gmail To Composed Equivalence Classes Mind Map





# Example: Gmail To Composed Equivalence Classes Mind Map





# Equivalence Classes

## Boundary Tests

### 2 Point Boundary

#### Min

- Right on Min
- Just Below Min

#### Max

- Right on Max
- Just Above Max

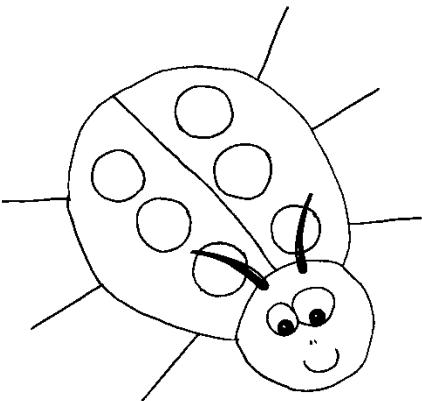
### 3 Point Boundary

#### Min

- Right On Min
- Just Below Min
- Just Above Min

#### Max

- Right On Max
- Just Below Max
- Just Above Max



# Boundary Testing To Infinity and Beyond ...

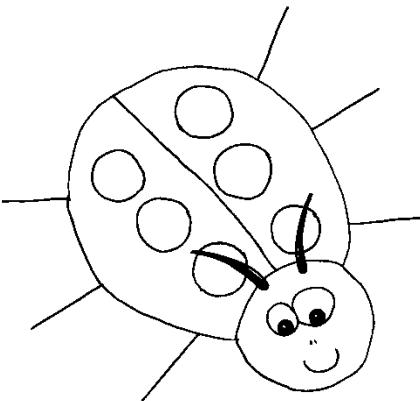
Robert Sabourin

President

AmiBug.Com, Inc.

Montreal, Canada

[rsabourin@amibug.com](mailto:rsabourin@amibug.com)



# Overview

## Introduction

## Traditional Boundary Risks

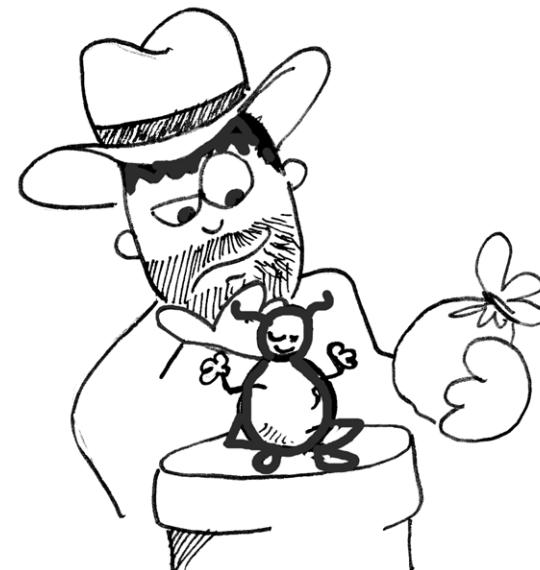
- Requirements
- Data Entry
- Processing

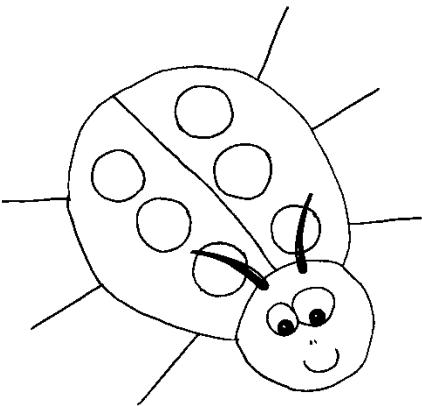
## Exploring Boundaries

- Testing Objectives
- Discover Variables
- Experimentation
- Analysis

## Stress Boundaries

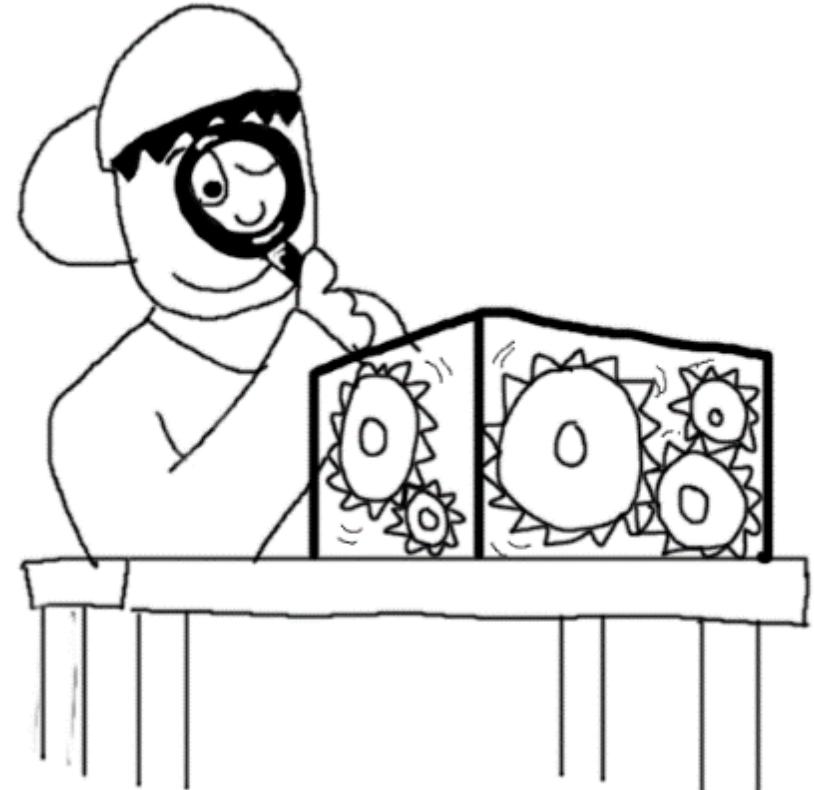
- Behavioral
- Performance
- Quality Factors

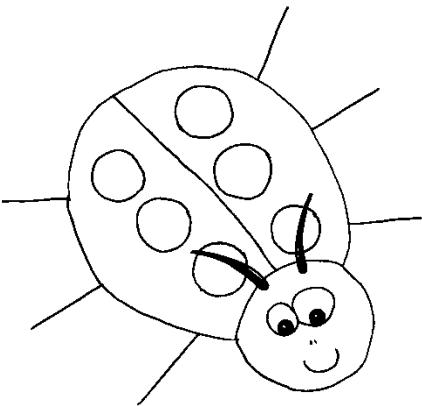




# Boundary Questions

- What is a boundary?
  - Many concepts
    - Extreme Values
    - Minimum
    - Maximum
    - Singularities
    - Discontinuities
    - Points at which behavior changes

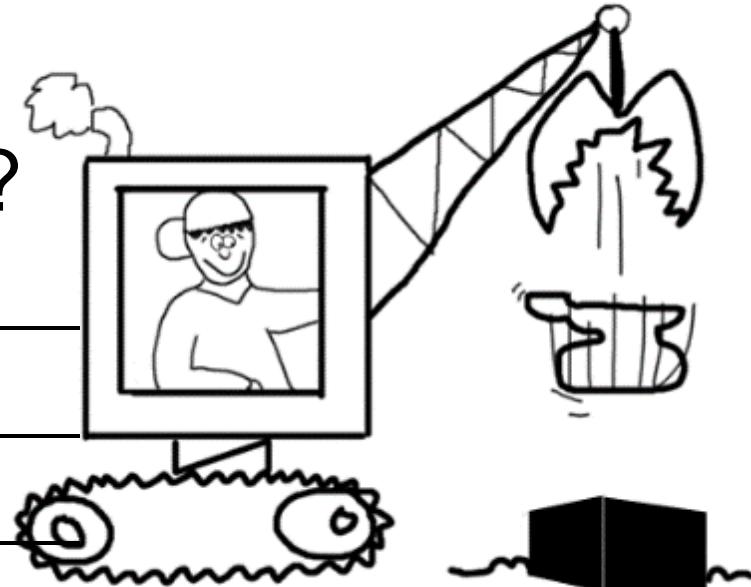


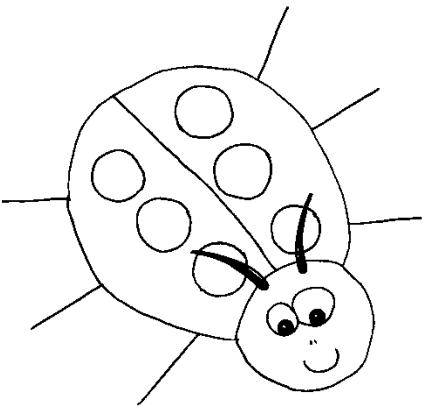


# Boundary Questions

- What is a boundary test?

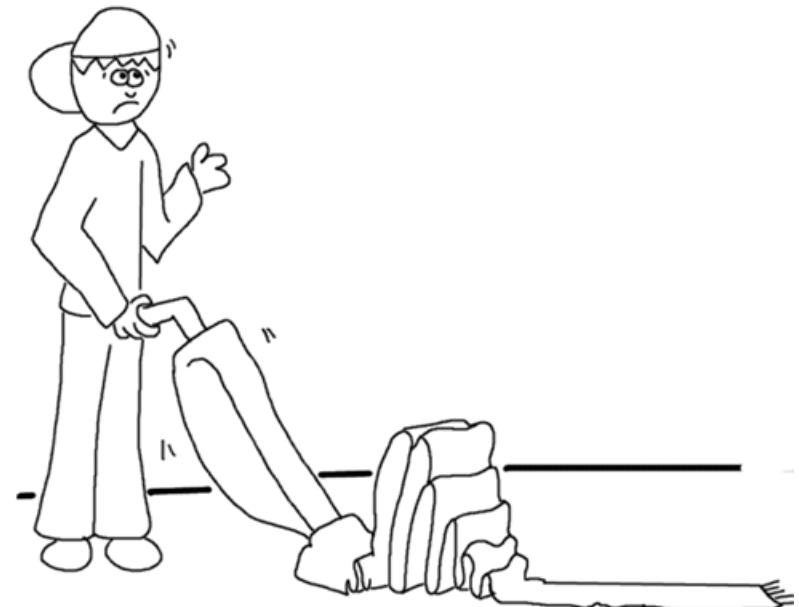
- 
- 
- 
- 

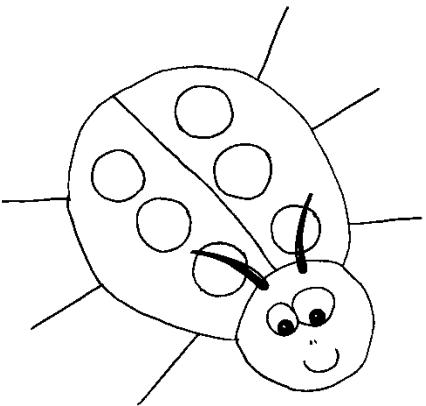




# Boundary Testing

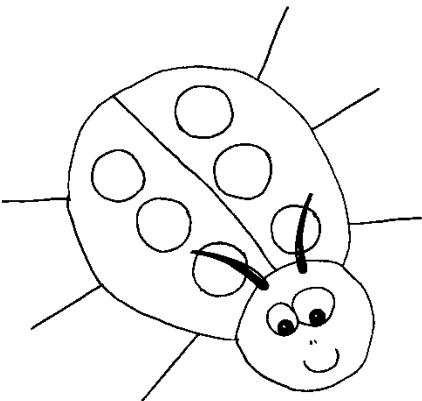
- Classes with continuous ranges of values
  - Test around extremes
  - Lower & upper boundaries
  - Edge conditions



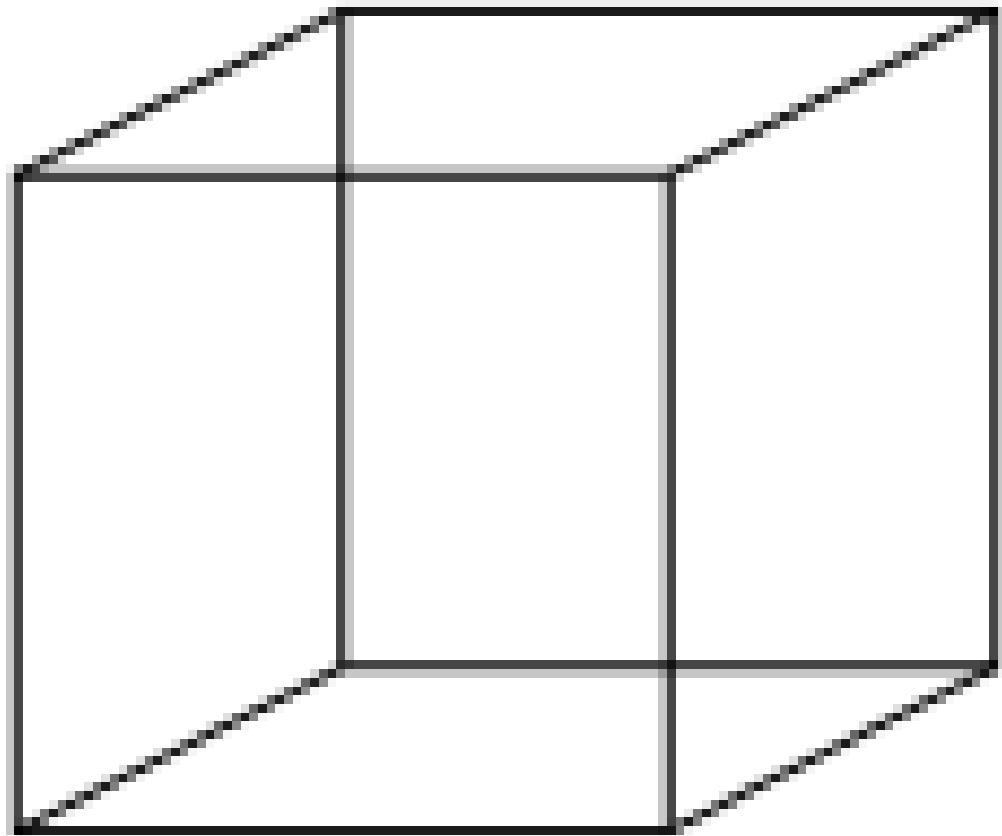


# Boundary Illusions



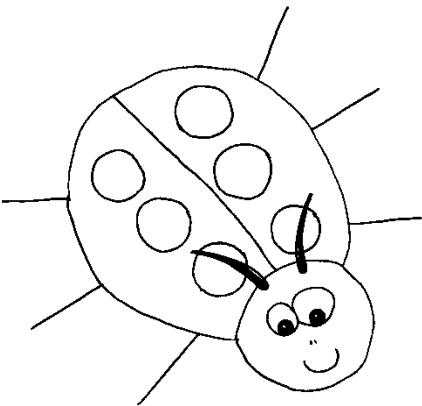


# Boundary Illusions



**Necker Cube**

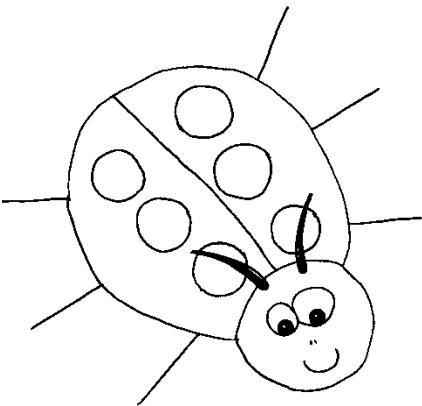
**Louis Albert Necker  
1832**



# Boundary Questions

- What is a boundary test?
  - Confirmatory test
    - Confirm behavior is as expected at boundaries of variables

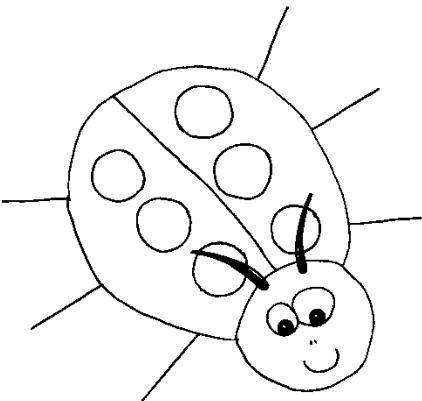




# Boundary Questions

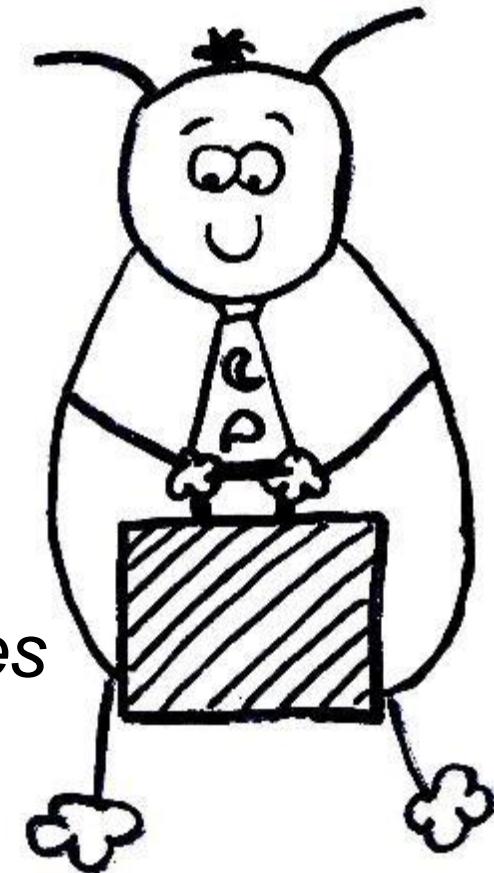
- What is a boundary test?
  - Exploratory discover ...
    - which variables influence behavior
    - the behavior as values are varied
    - values which do not cause behaviors to change

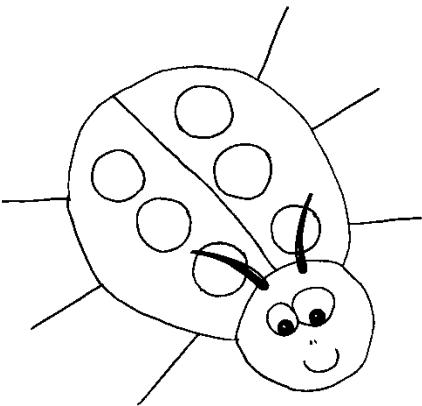




# Boundary Questions

- Why are boundary tests important?
  - Experience
  - Knowledge
  - *Boundaries values are sometimes considered as the **best representative values** in an equivalence class*

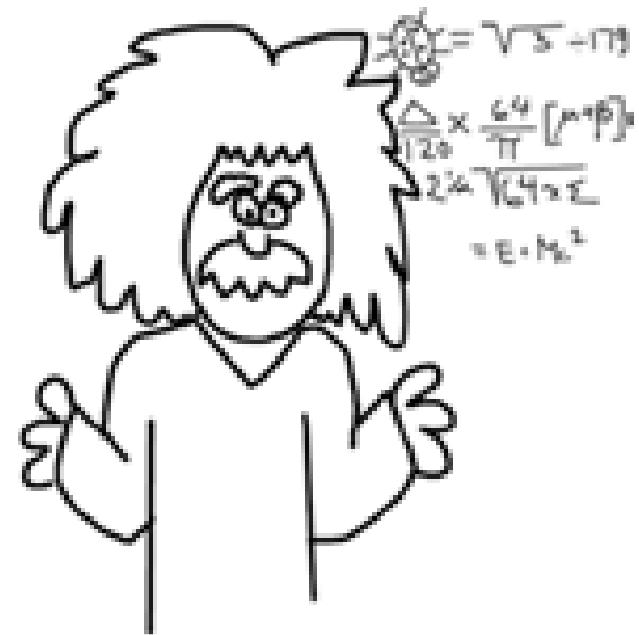


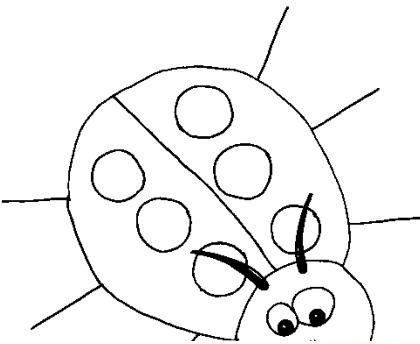


# Taking AIM

## Testing Approaches

- Equivalence Classes
- Usage Scenarios
- Quality Factors



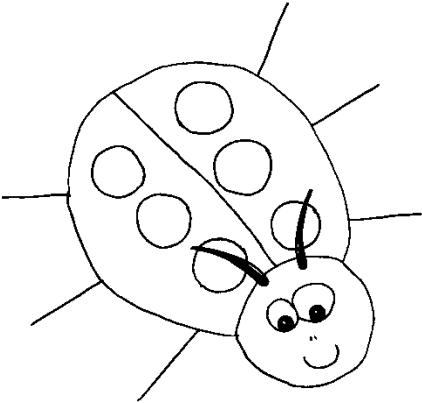


# Multiple Boundaries

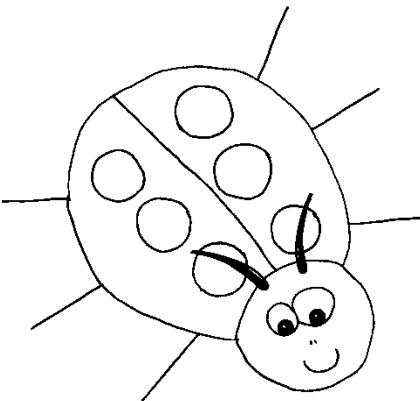
## COMMERCIAL & OFFICIAL SIZE ENVELOPES

						(5 x 11 $\frac{1}{2}$ ) #14
						(4 $\frac{3}{4}$ x 11) #12
						(4 $\frac{1}{2}$ x 10 $\frac{3}{8}$ ) #11
						(4 $\frac{1}{8}$ x 9 $\frac{1}{2}$ ) #10
HOW TO USE:		(3 $\frac{1}{2}$ x 6) #6 1/4	#6 3/4 (3 $\frac{5}{8}$ x 6 $\frac{1}{2}$ )	#7 (3 $\frac{3}{4}$ x 6 $\frac{1}{2}$ )	#7 3/4 or Monarch (3 $\frac{1}{8}$ x 7 $\frac{1}{2}$ )	#8 5/8* or Check (3 $\frac{5}{8}$ x 8 $\frac{5}{8}$ )
Position corner of insert against dash rule. (This allows space for easy inserting.) If address area on insert appears within window area, you can use the standard envelope indicated. Special size envelopes and/or windows available on order.						*Also available in standard window size 1" x 4". Position from left, 1"; Position from bottom 3/4".

# Multiple Boundaries



Category	Length	Width	Thickness	Weight	
Standard					
	max.	245 mm	156 mm	5 mm	50 g
Standard Envelope/Self-mailer					
	min.	140 mm	90 mm	0.18 mm	3 g
	max.	235 mm	120 mm	5 mm	50 g
Cards or Postcards					
	min.	140 mm	90 mm	0.18 mm	3 g
	max.	380 mm	270 mm	20 mm	500 g
Other (Non-standard and Oversize) Envelope/Self-mailer/Card					
	min.	140 mm	90 mm	0.18 mm	10 g



# Example Precision Requirements

**Table 1.** Readings as a function of accuracy

Input Voltage	Range of Readings within the Accuracy Specification
0 V	-1 mV to +1 mV
5 V	4.994 V to 5.006 V ( $\pm 6$ mV)
10 V	9.989 V to 10.011 V ( $\pm 11$ mV)

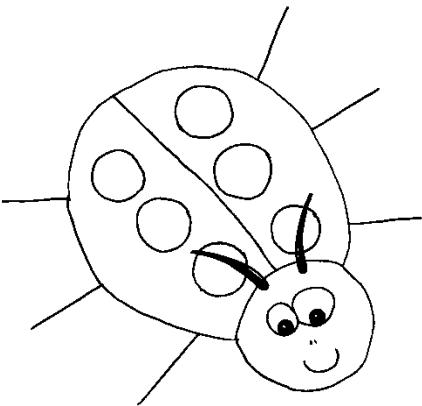
**Conditions:** input 0-10 V, Accuracy =  $\pm(0.1\% \text{ of input} + 1 \text{ mV})$

## Precision

Precision describes the reproducibility of the measurement. For example, measure a steady state signal many times. In this case if the values are close together then it has a high degree of precision or repeatability. The values do not have to be the true values just grouped together. Take the average of the measurements and the difference is between it and the true value is accuracy.

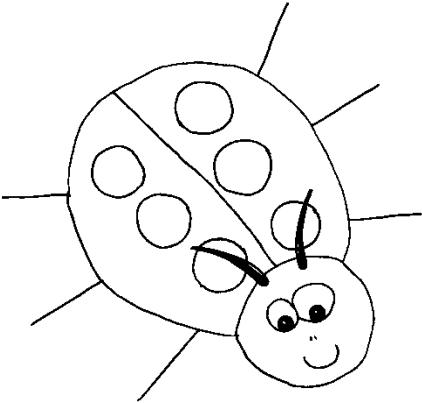
MC Computing

<https://www.mccdaq.com/TechTips/TechTip-1.aspx>



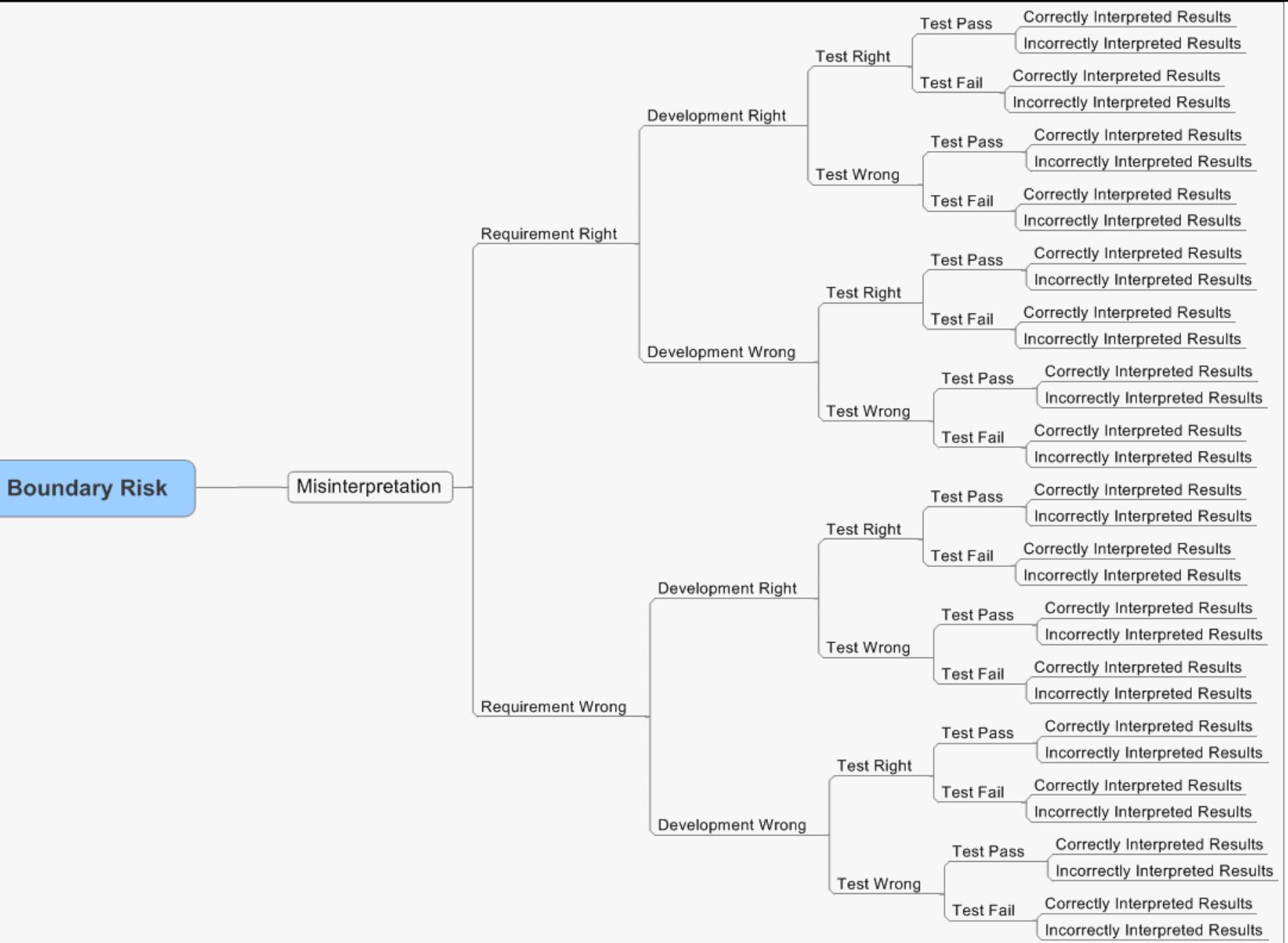
# Precision Machine Epsilon

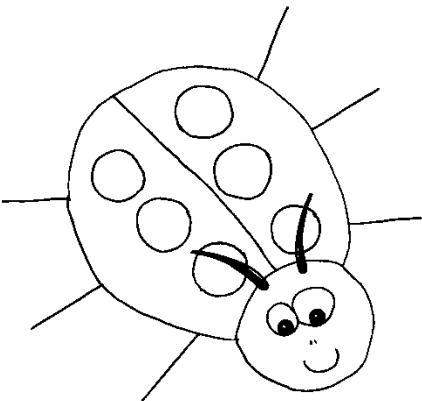
```
epsilon = 1.0;  
  
while (1.0 + 0.5 * epsilon) ≠ 1.0:  
    epsilon = 0.5 * epsilon
```



# Precision Machine Epsilon

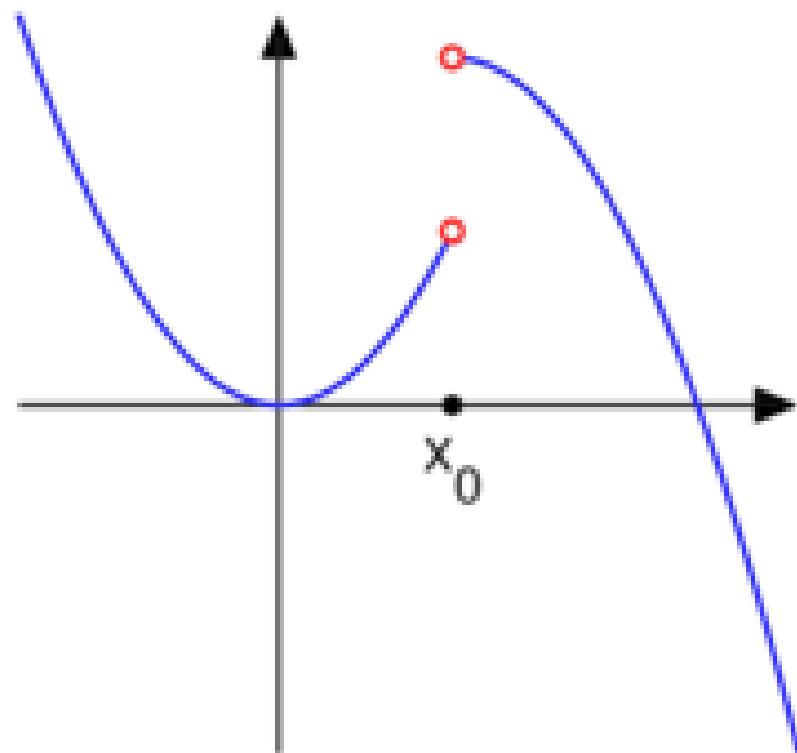
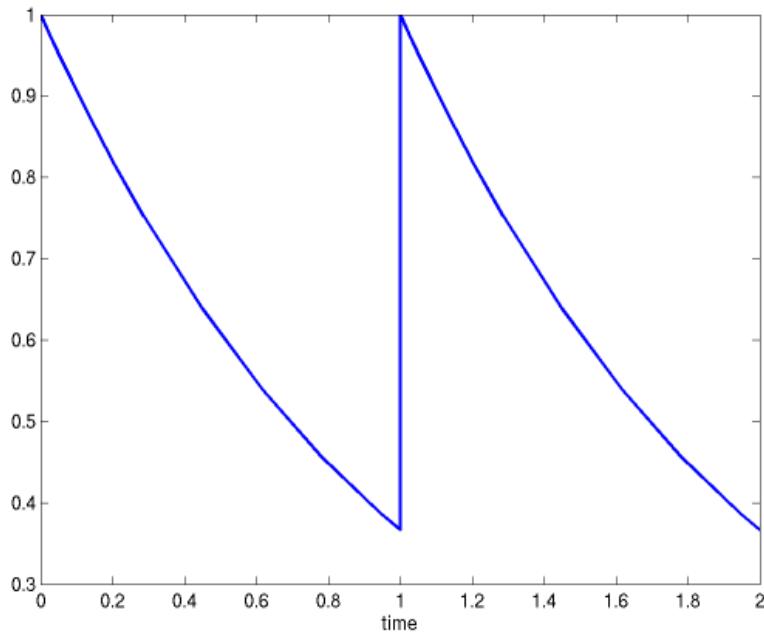
IEEE 754 - 2008	Common name	C++ data type	Base $b$	Precision $p$	Machine epsilon <sup>[a]</sup> $b^{-(p-1)} / 2$	Machine epsilon <sup>[b]</sup> $b^{-(p-1)}$
binary16	half precision	N/A	2	11 (one bit is implicit)	$2^{-11} \approx 4.88e-04$	$2^{-10} \approx 9.77e-04$
binary32	single precision	float	2	24 (one bit is implicit)	$2^{-24} \approx 5.96e-08$	$2^{-23} \approx 1.19e-07$
binary64	double precision	double	2	53 (one bit is implicit)	$2^{-53} \approx 1.11e-16$	$2^{-52} \approx 2.22e-16$
	extended precision, long double	_float80 <sup>[1]</sup>	2	64	$2^{-64} \approx 5.42e-20$	$2^{-63} \approx 1.08e-19$
binary128	quad(ruple) precision	_float128 <sup>[1]</sup>	2	113 (one bit is implicit)	$2^{-113} \approx 9.63e-35$	$2^{-112} \approx 1.93e-34$
decimal32	single precision decimal	_Decimal32 <sup>[2]</sup>	10	7	$5 \times 10^{-7}$	$10^{-6}$
decimal64	double precision decimal	_Decimal64 <sup>[2]</sup>	10	16	$5 \times 10^{-16}$	$10^{-15}$
decimal128	quad(ruple) precision decimal	_Decimal128 <sup>[2]</sup>	10	34	$5 \times 10^{-34}$	$10^{-33}$

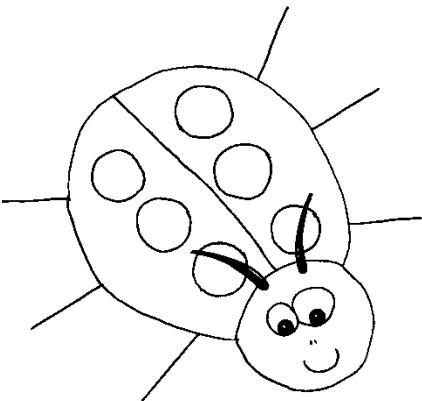




# Boundary

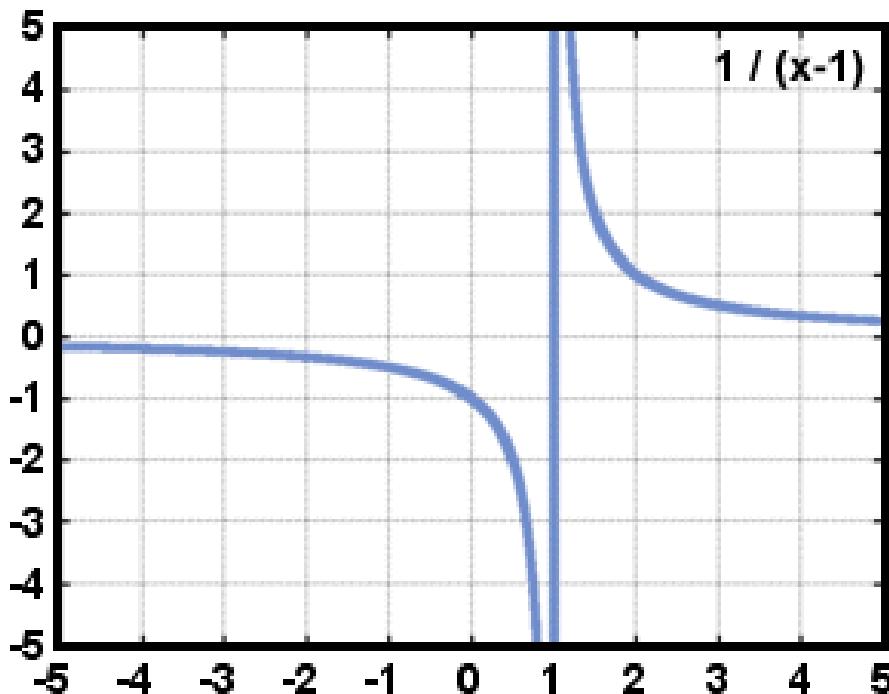
- Discontinuity

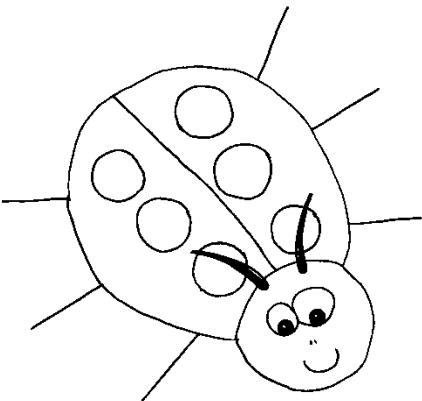




# Boundary

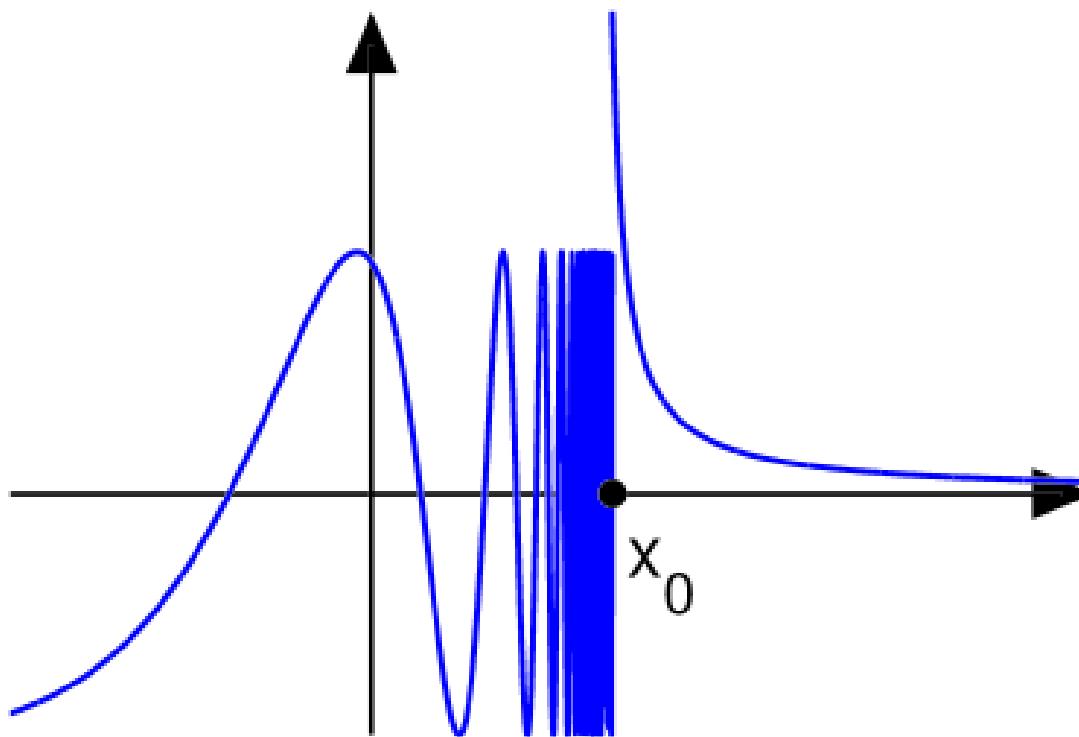
- Singularity

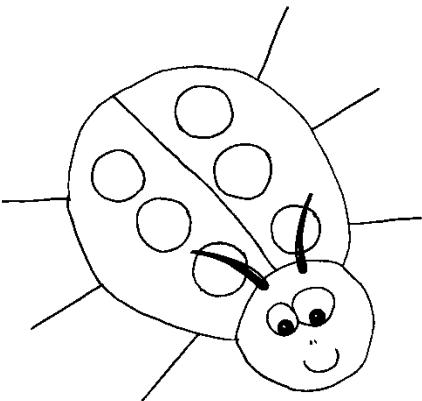




# Boundary

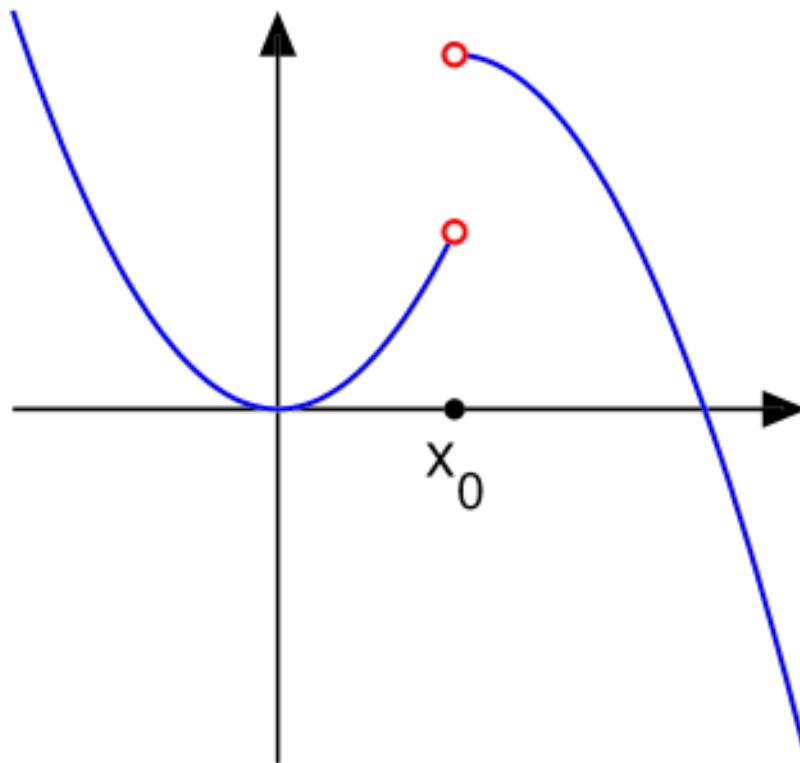
- Behavioral Change

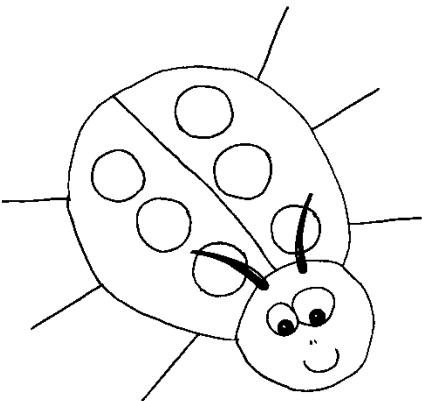




# Boundary

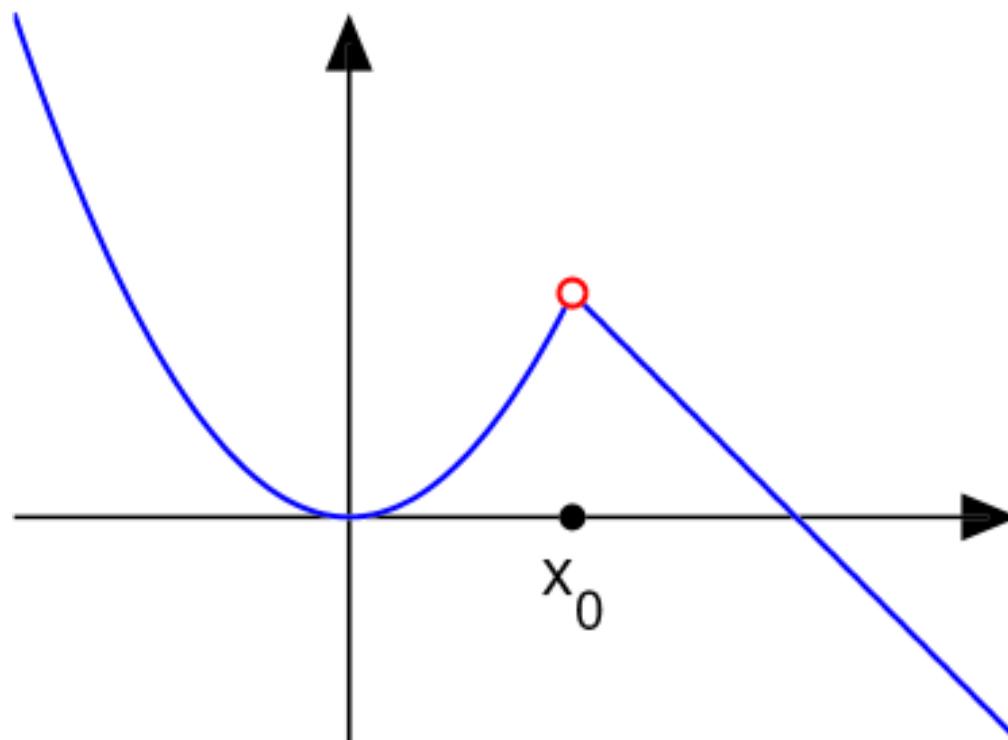
- Jump Discontinuity

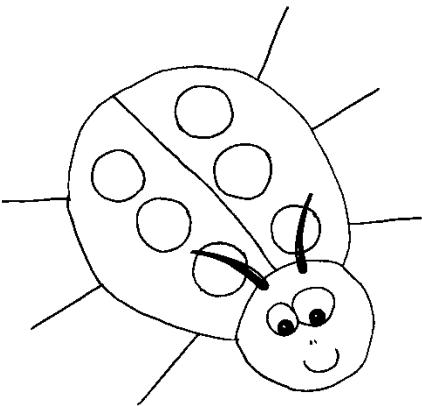




# Boundary

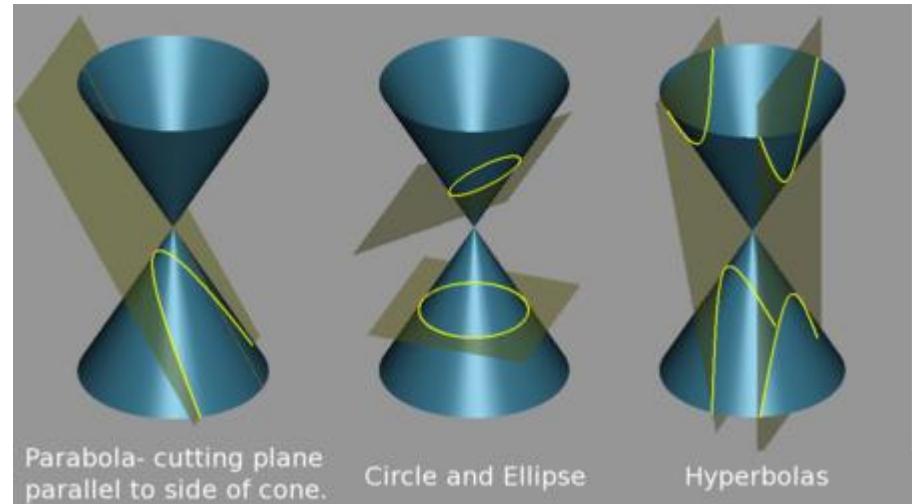
- Removable Discontinuity

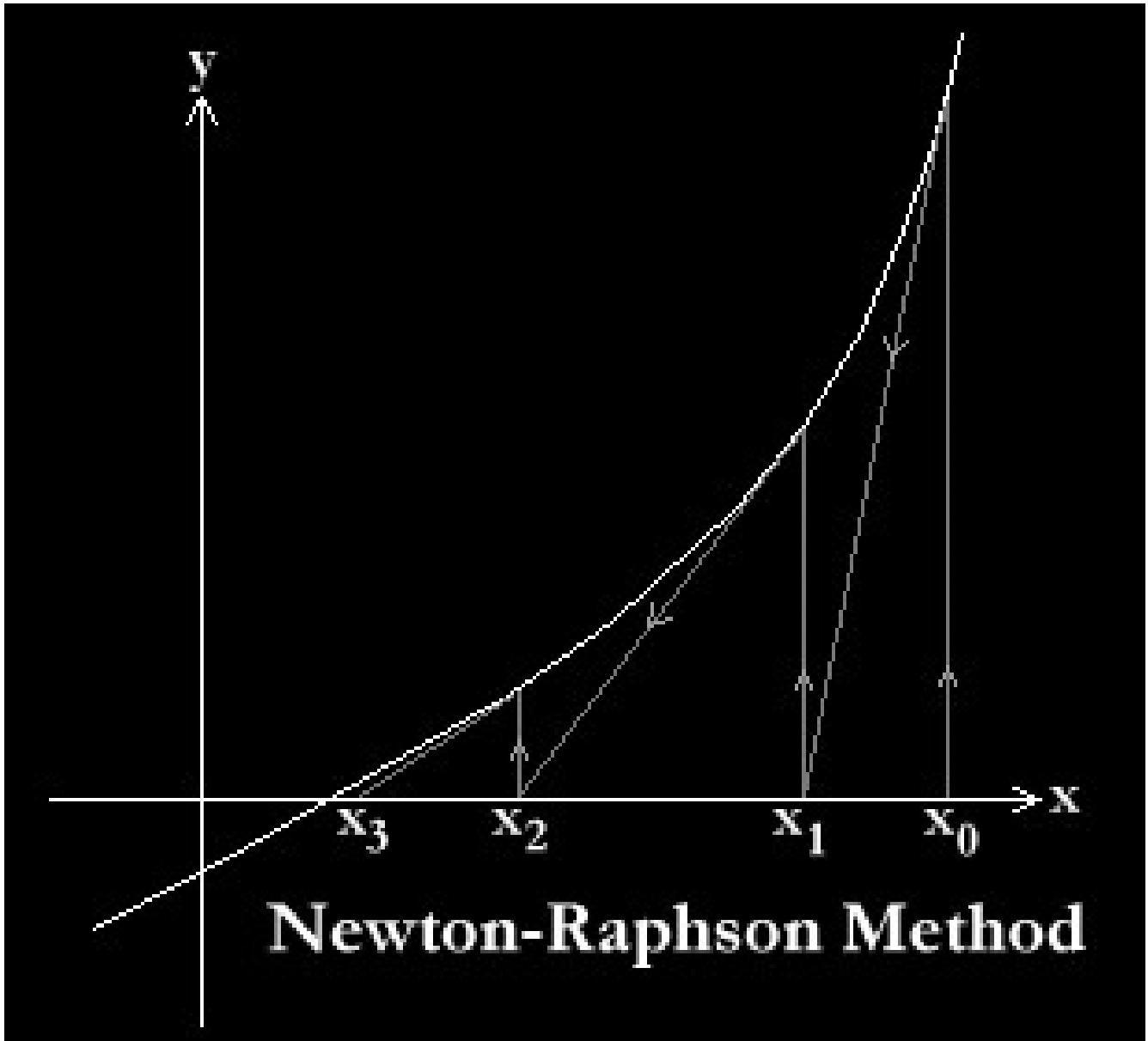
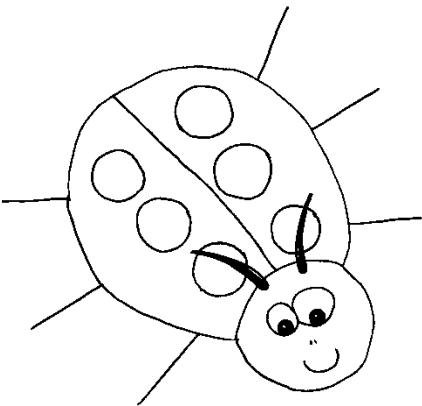


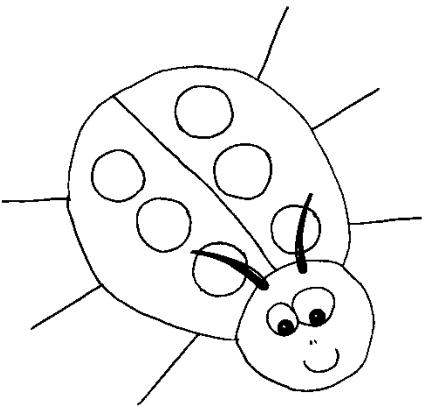


# Boundary

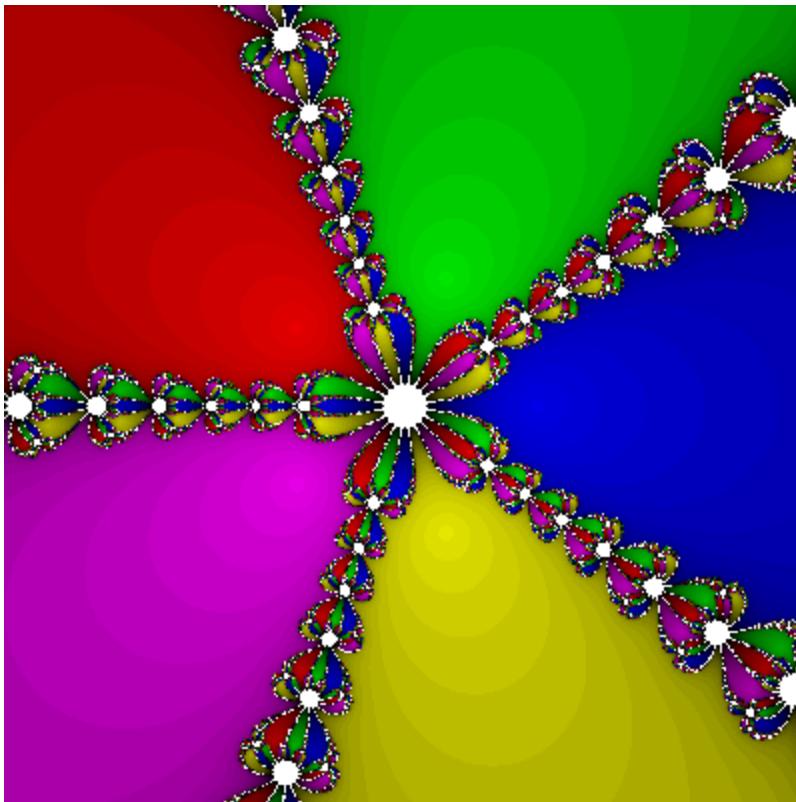
- Conic Sections
- Intersect a plane with a cone
  - Parabola
  - Circle
  - Ellipse
  - Hyperbola



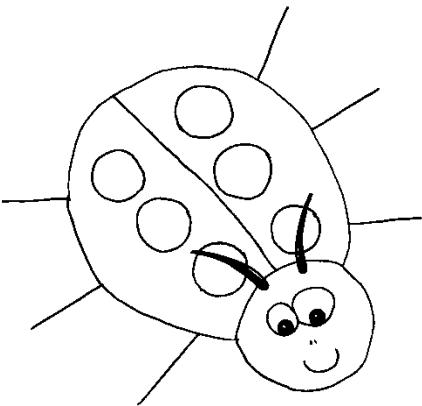




# Boundary

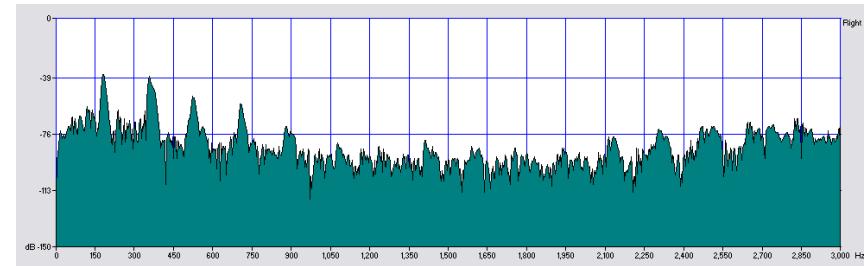
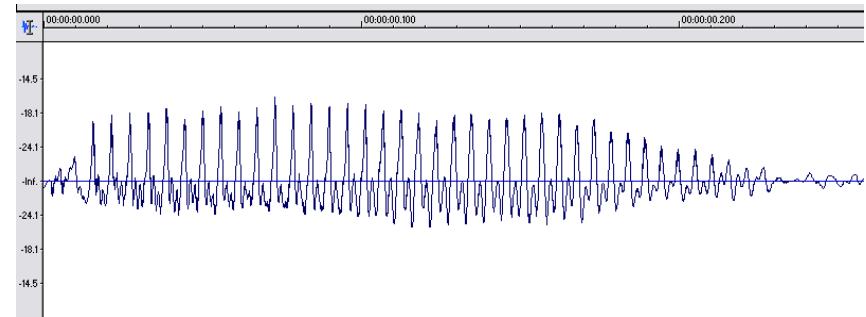


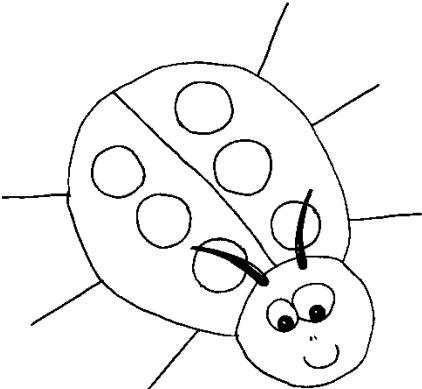
Fractal Roots  
Polynomial:  $x^{**5} - 1$



# Boundary

- Time Domain
- Frequency Domain

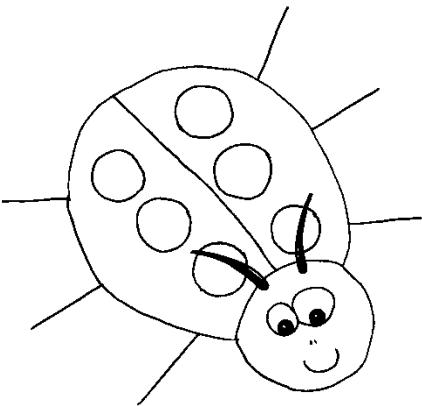




# Boundary

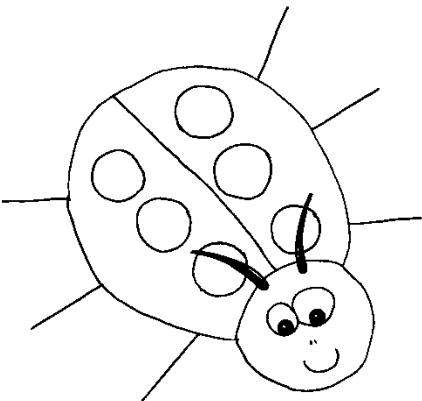
- Political Geography





# Boundary

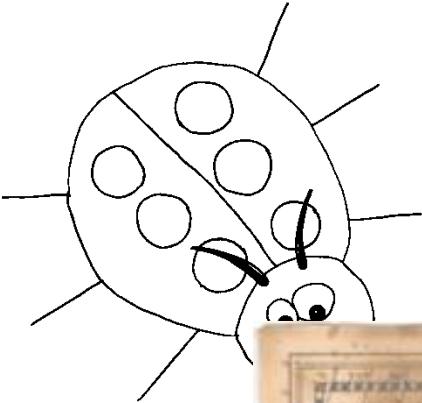




# Boundary

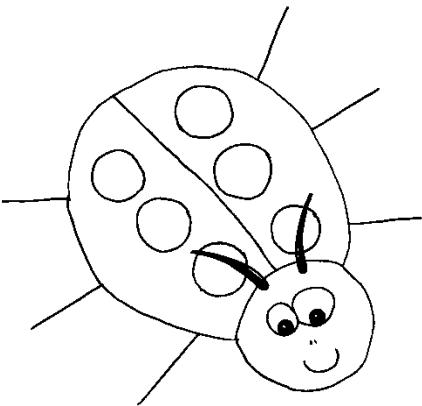
- Physical Geography





# Boundary

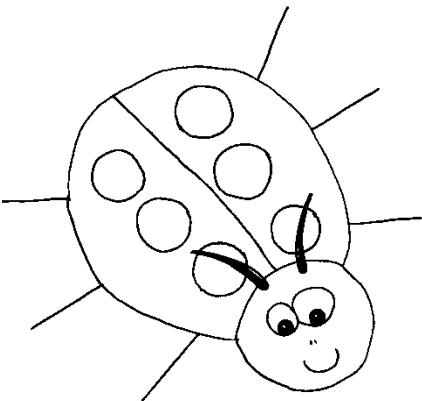




# Boundary

- Going over the edge

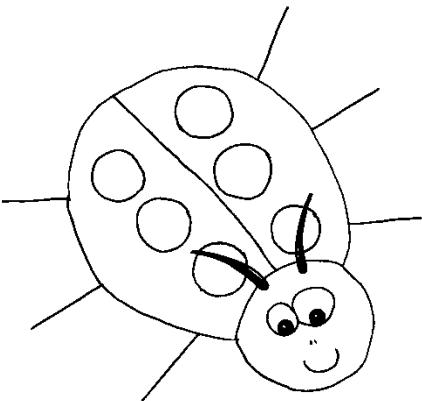




# Boundary

- Off the edge of the earth

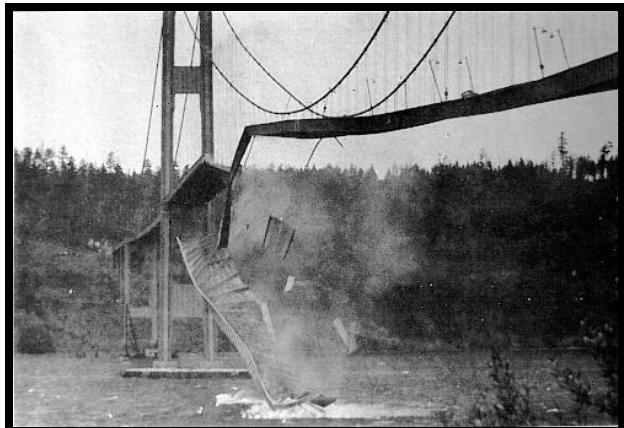
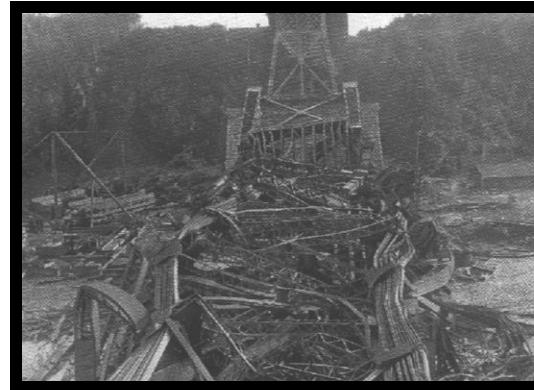




# Analysis & Design

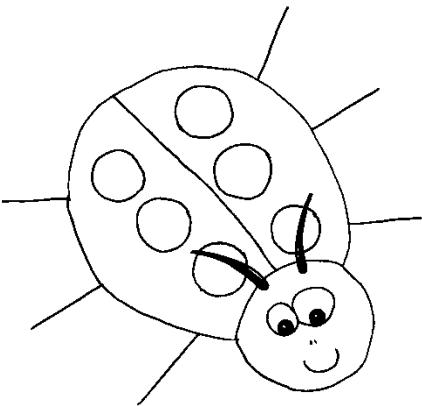
## Quebec City Bridge, 1916

- *Construction collapse*
- *Stress due to scale*



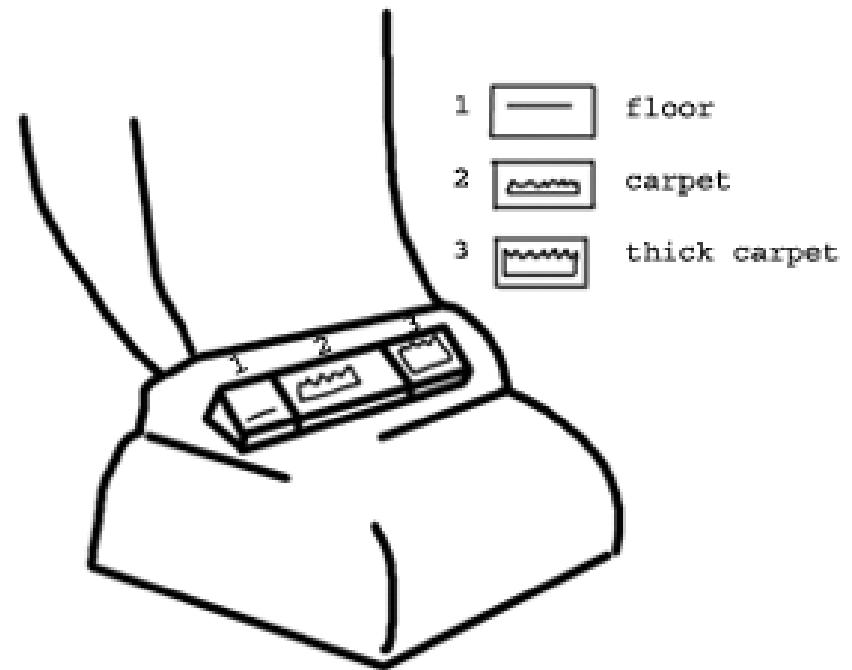
## Tacoma Narrows, 1940

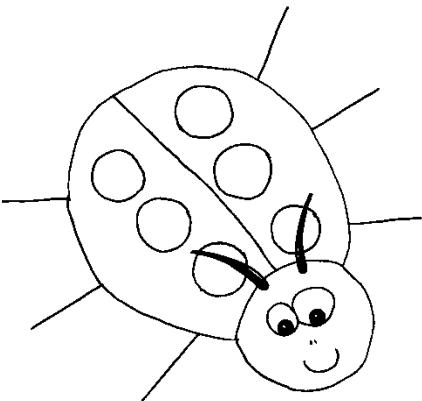
- *Collapse during normal operation*
- *Stress due to instability*
- *Wind was only 42 mph*



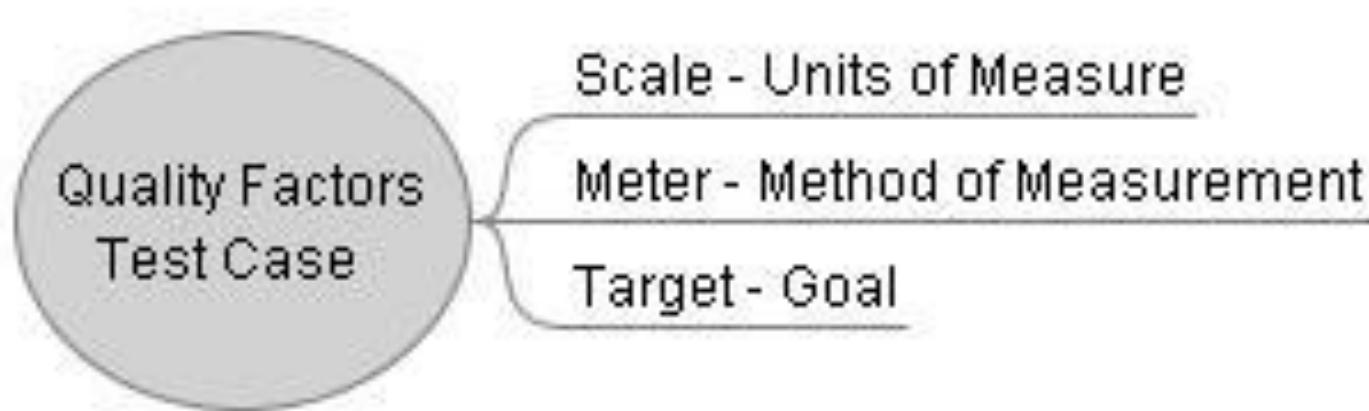
# Taking AIM

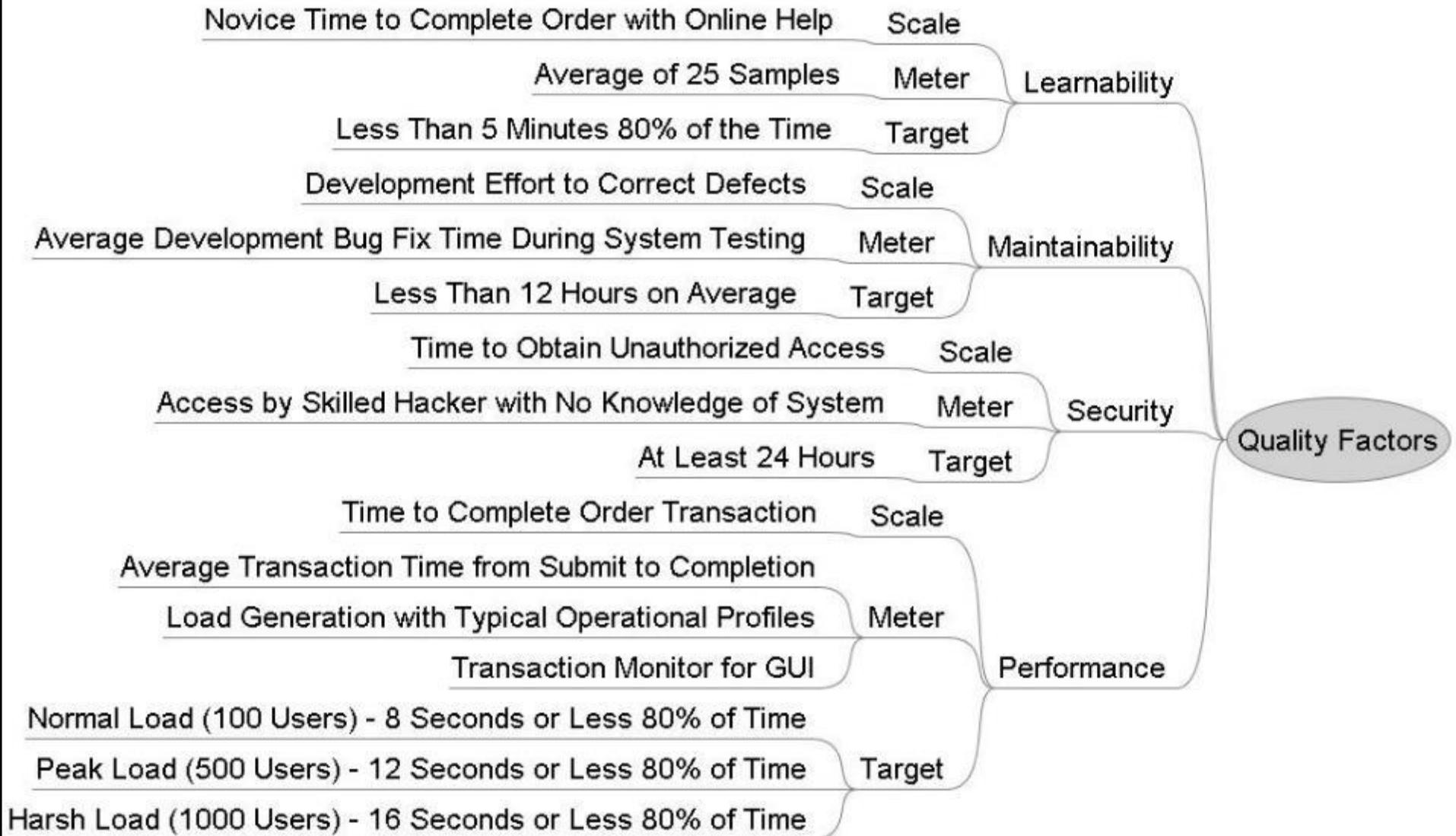
- Quality Factors
  - Scale
  - Meter
  - Goal



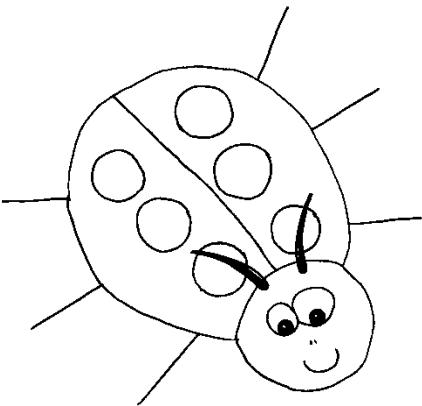


# Taking AIM





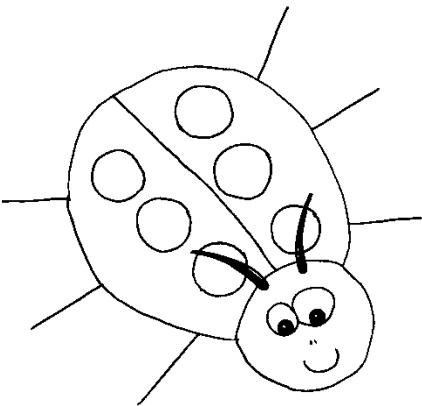
# Taking AIM



# Boundaries

- Limits of authority

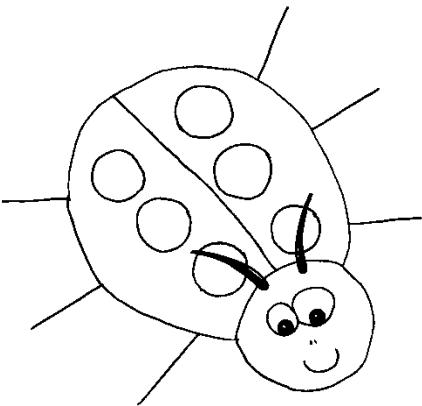




# Boundary

- Limits of patience

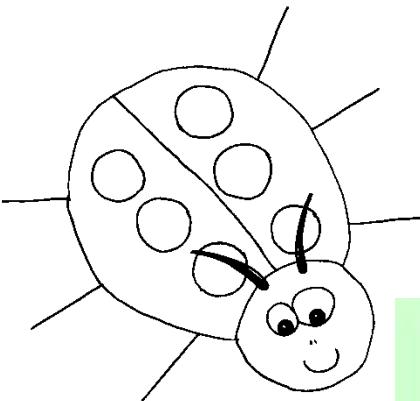




# Boundary

- Visual acuity
  - Are these two points the same



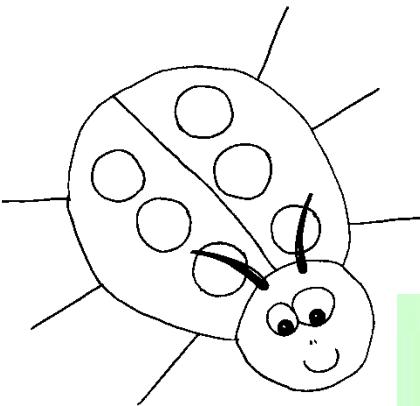


# Boundary

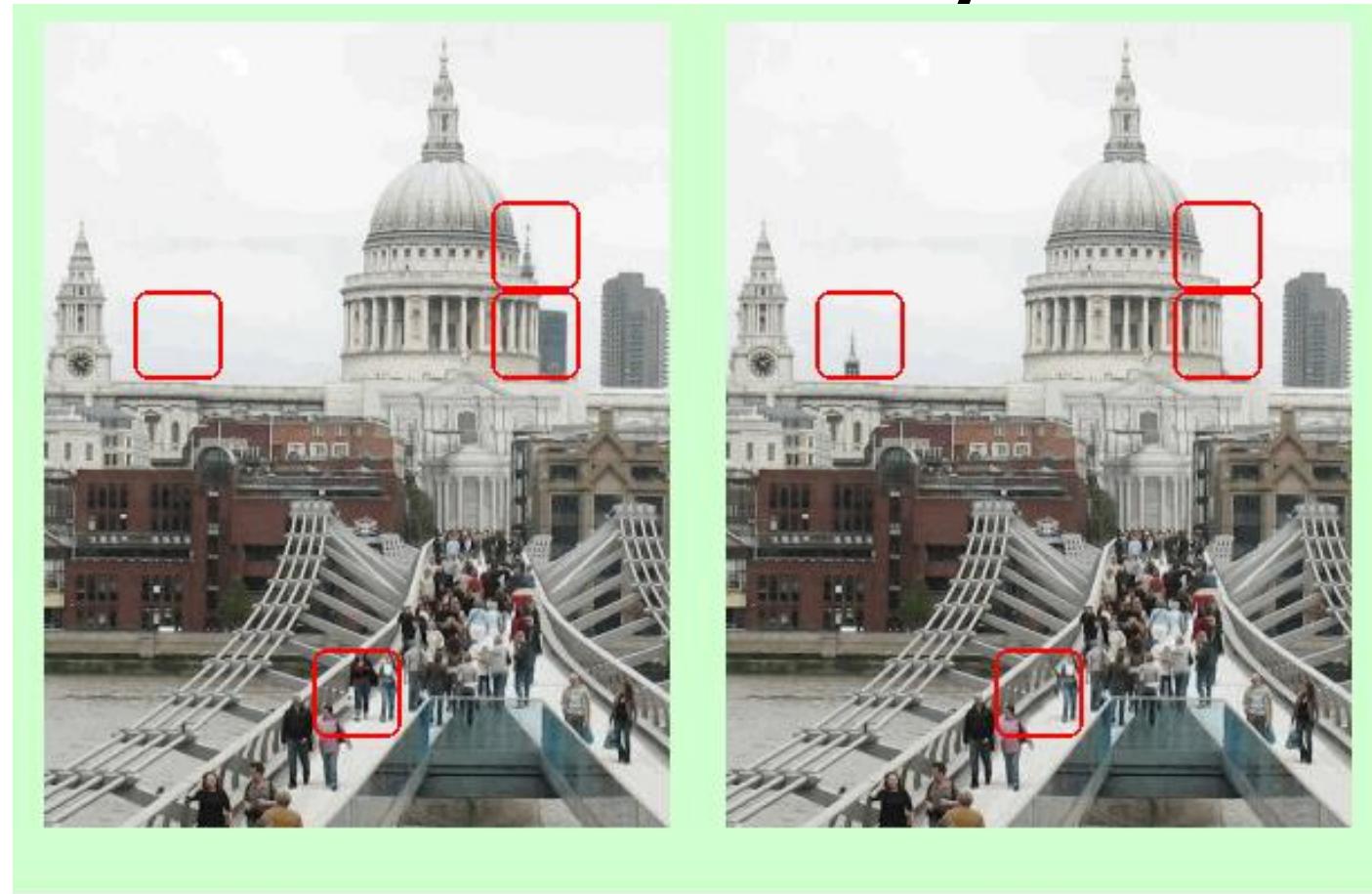


**Find 4 differences.**

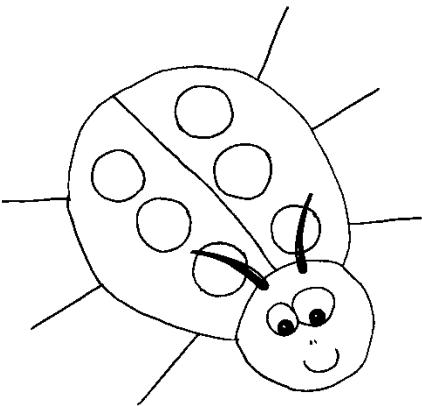
**Give up**



# Boundary

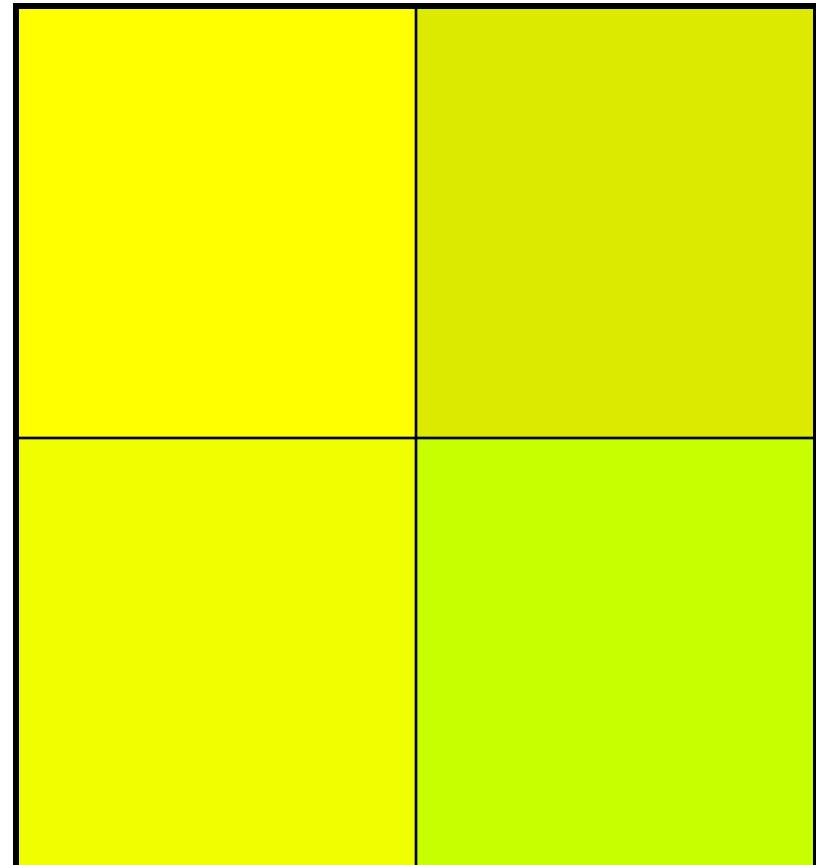


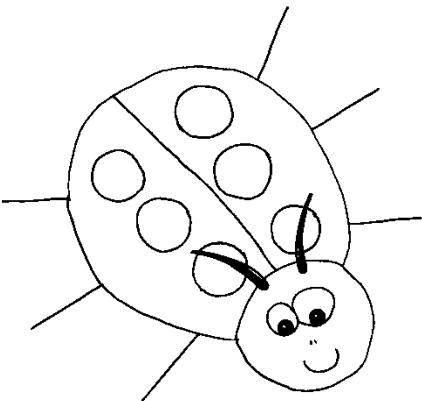
Found 4 differences.



# Boundary

- Color
  - Correctly name shades of yellow

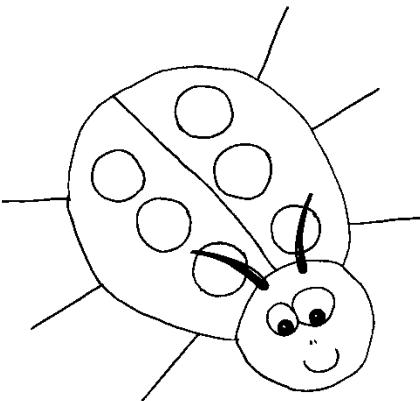




# Boundary

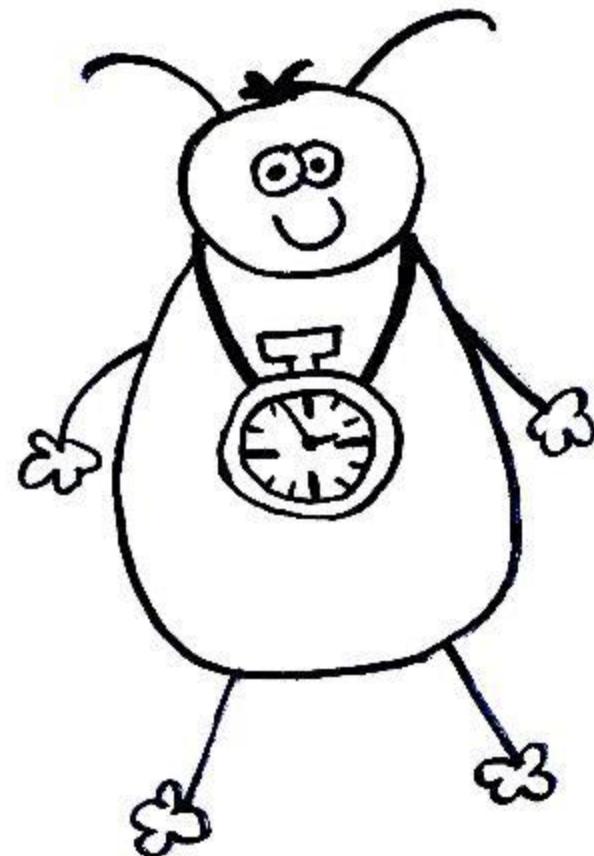
- Phonetic Similarity
- **Conscious, Conscience**
  - conscious = adjective meaning awake, perceiving: Despite a head injury, the patient remained **conscious**.
  - conscience = noun meaning the sense of obligation to be good: Chris wouldn't cheat because his **conscience** wouldn't let him.

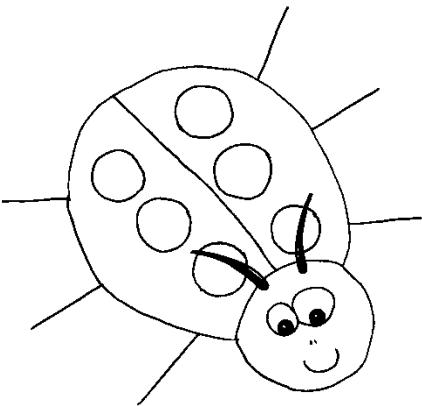




# Boundary

- Delay timing
  - Pauses and delays in the middle of transactions

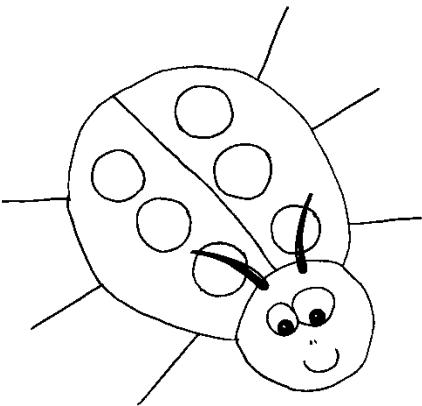




# Boundary

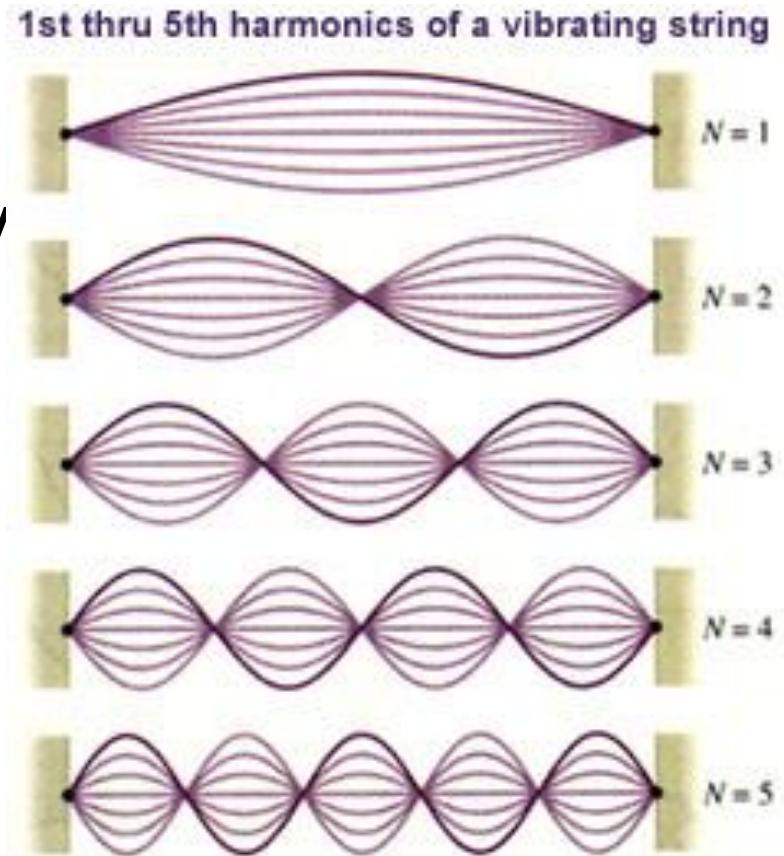
- Security
  - The ethical hacker
  - How long to break in?

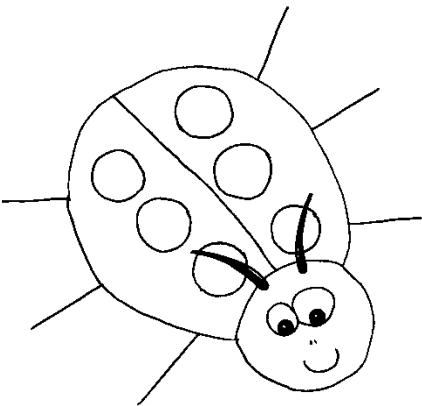




# Boundary

- Resonant frequency



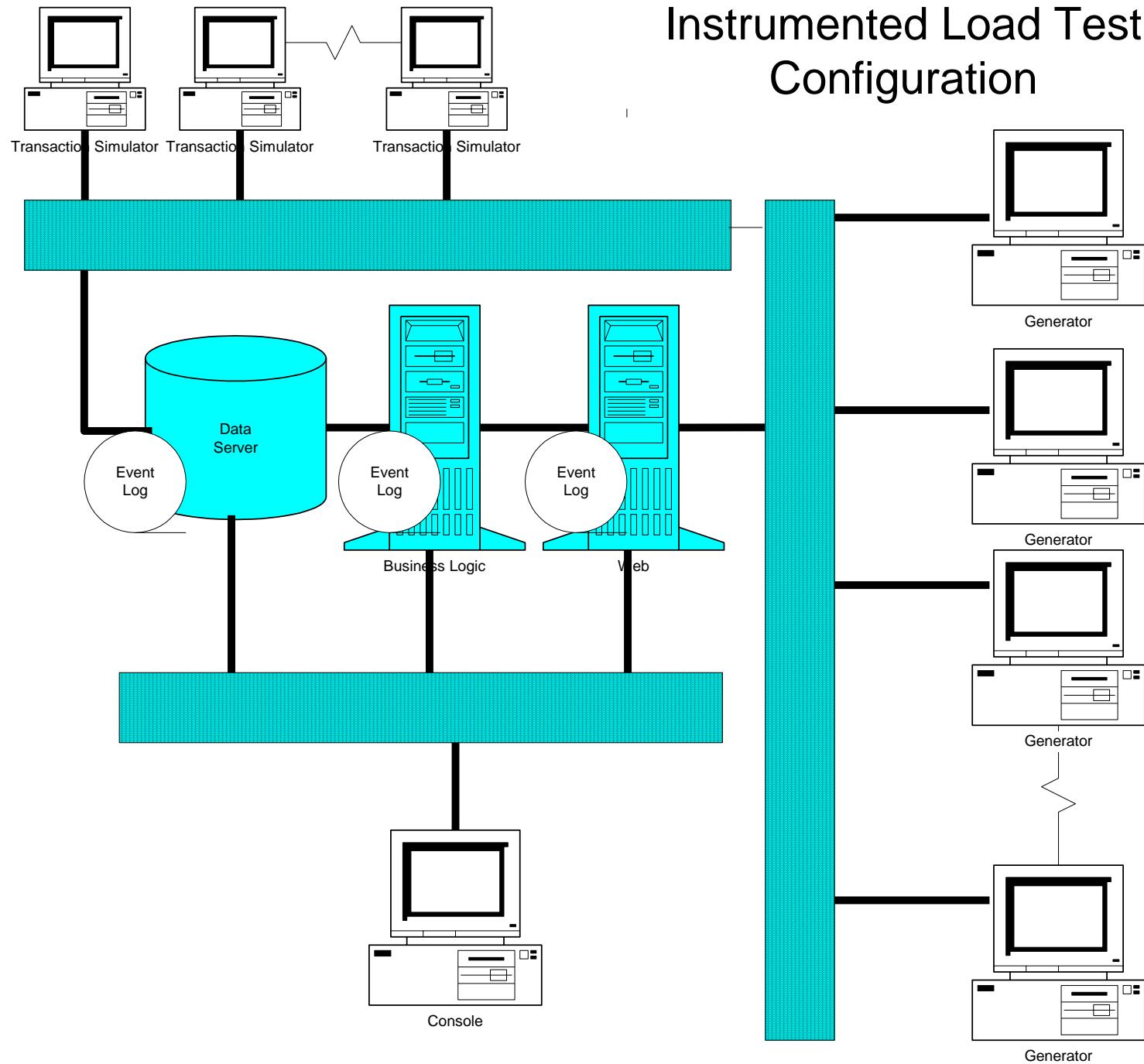


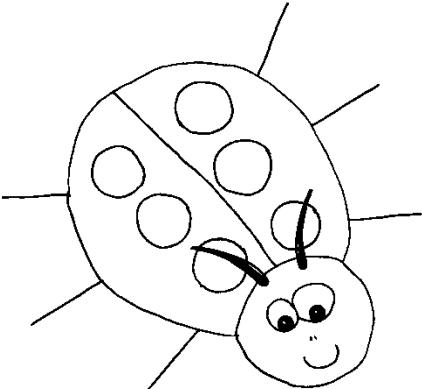
# Boundary

- Toy train set
- Which variables lead to a crash?

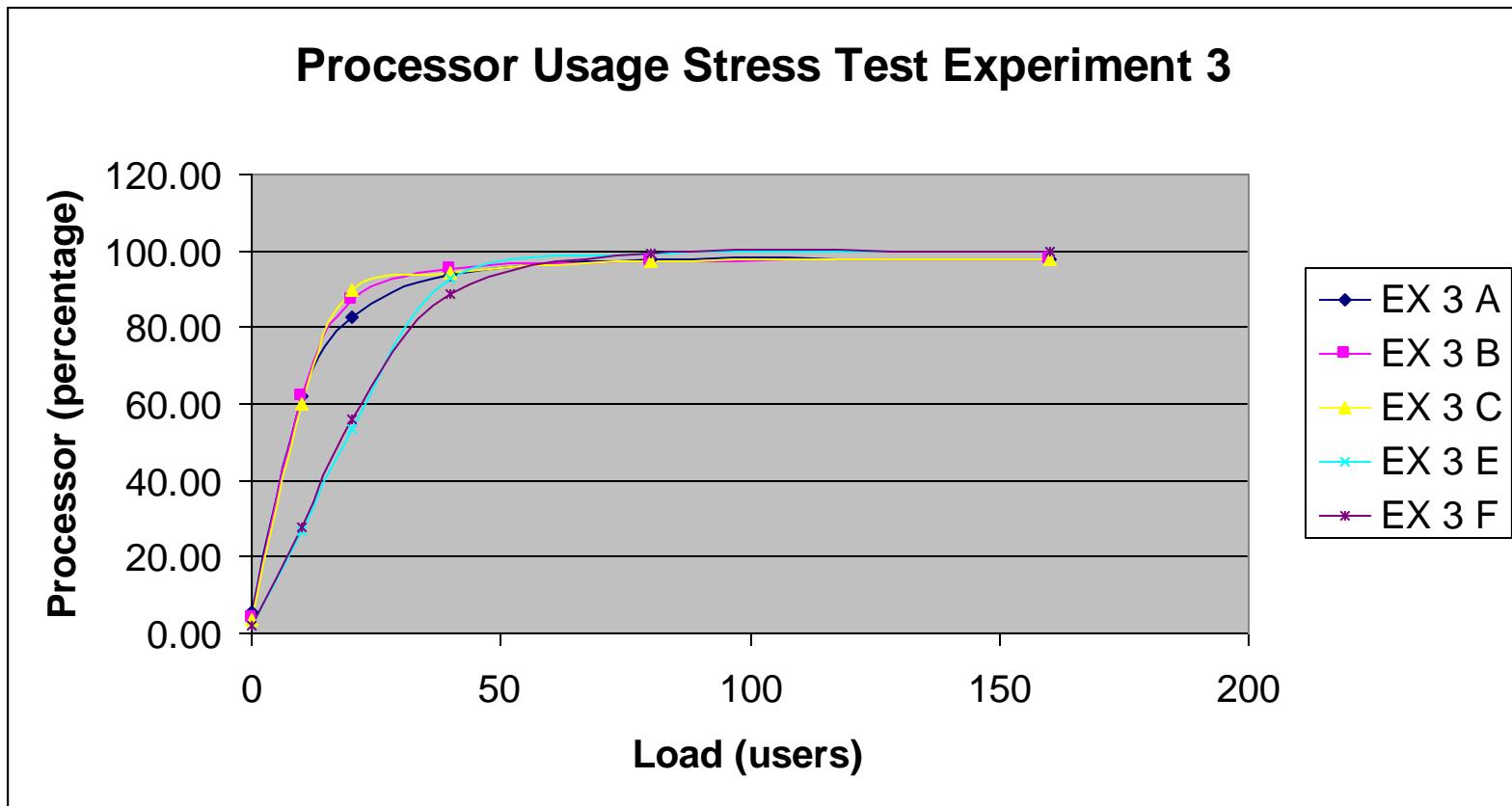


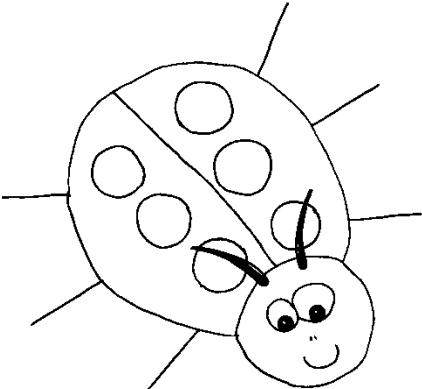
# Instrumented Load Test Configuration



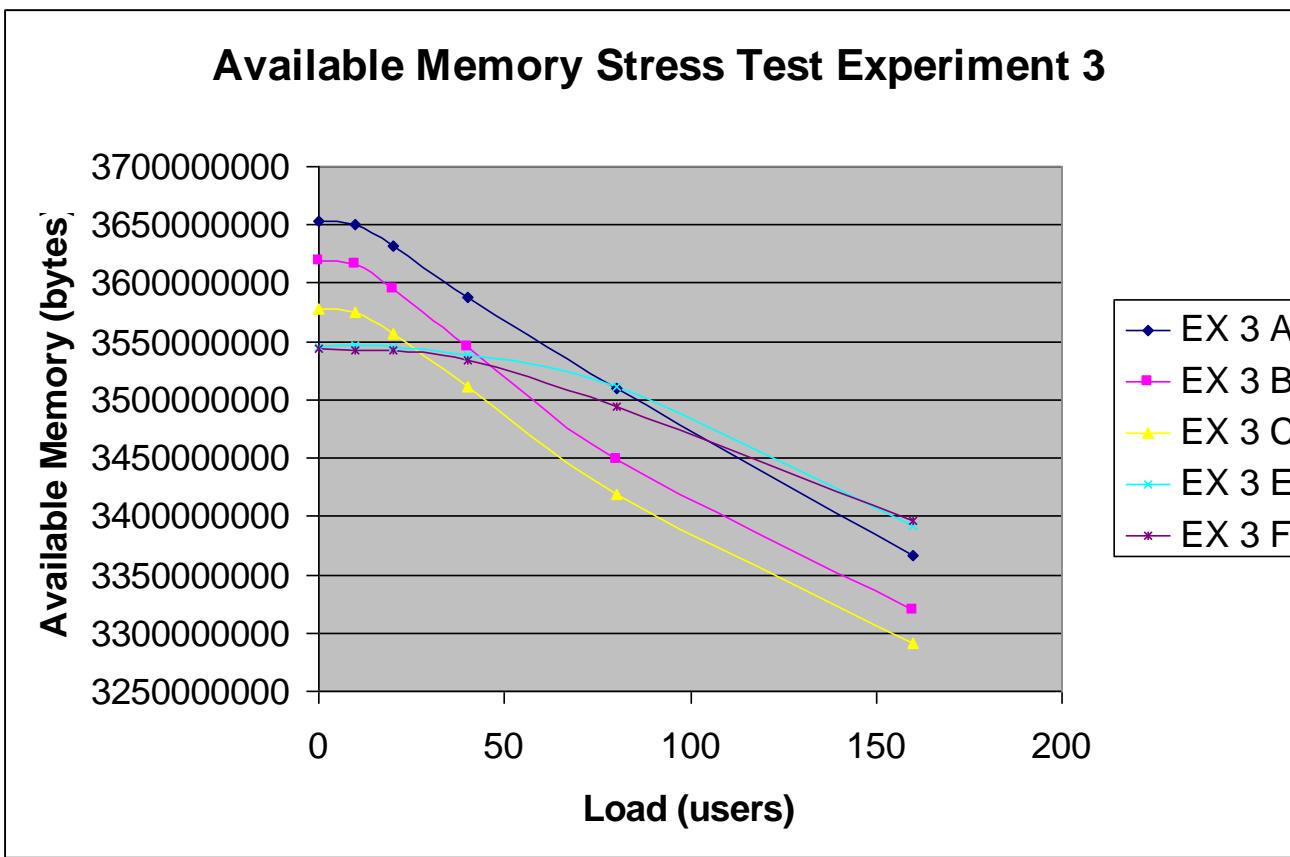


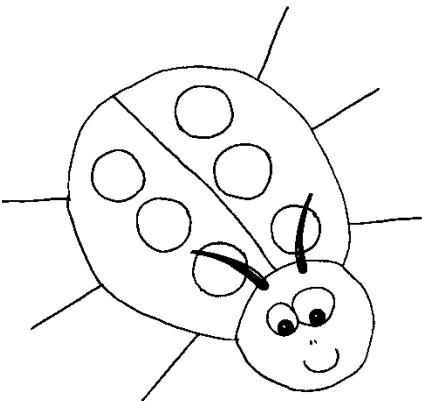
# Performance Testing Experiments



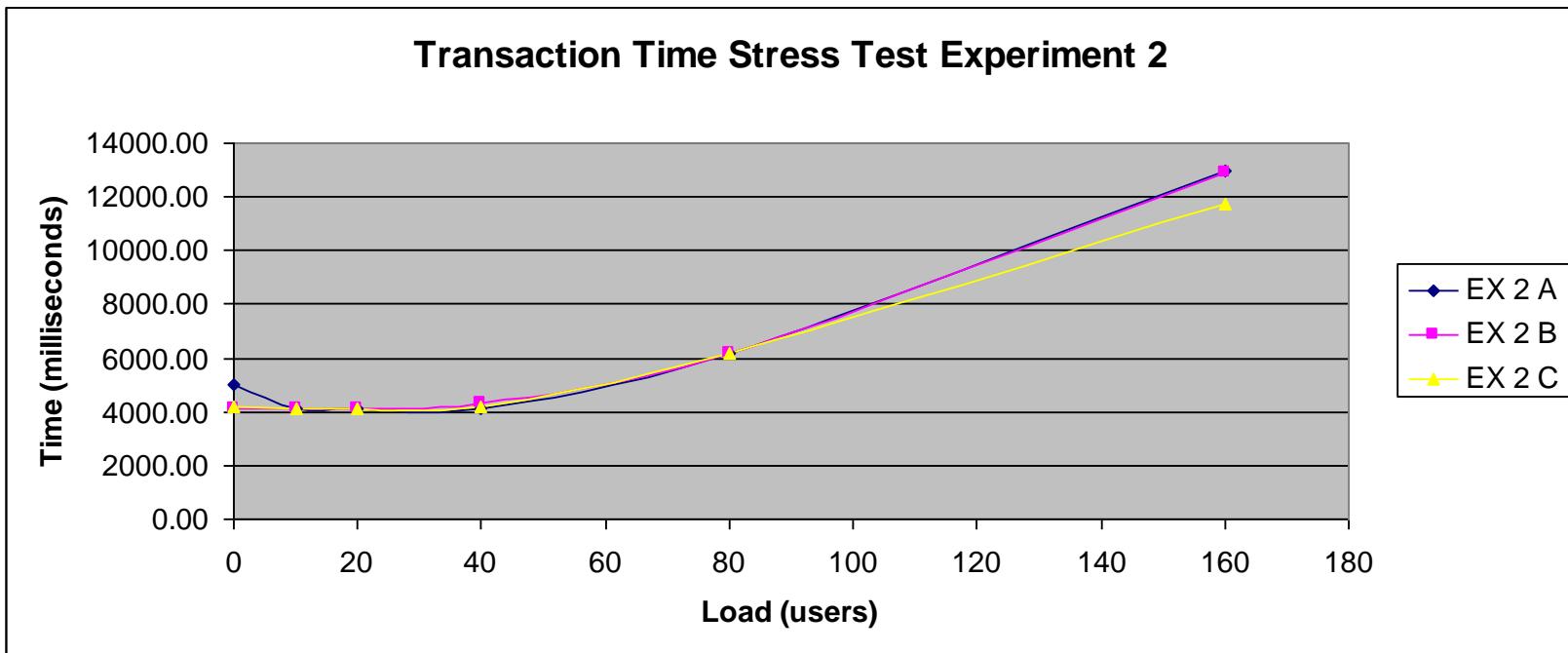


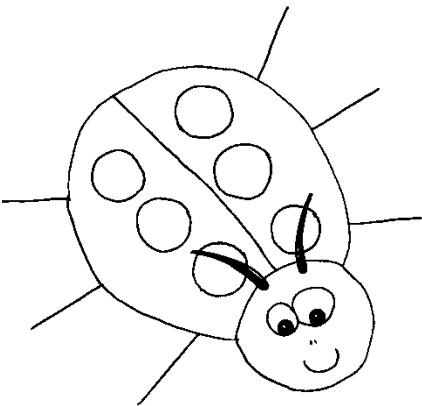
# Performance Testing Experiments





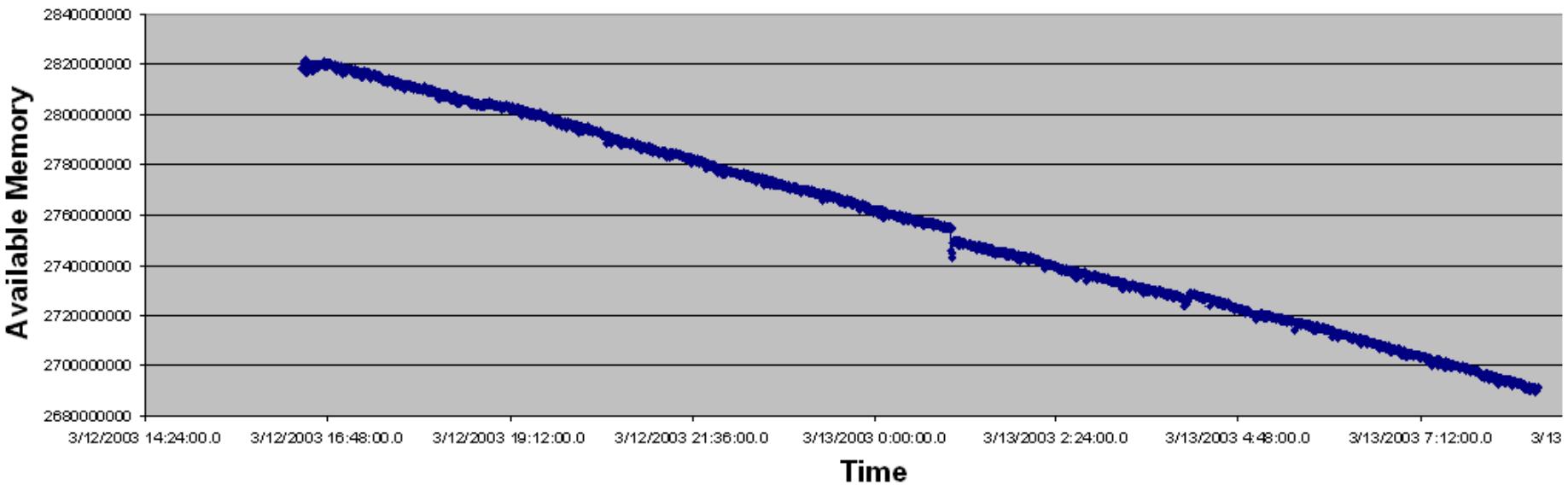
# Performance Testing Experiments

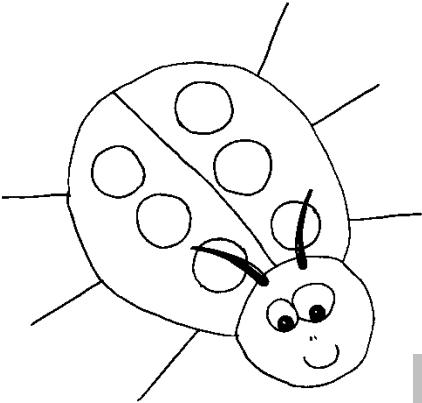




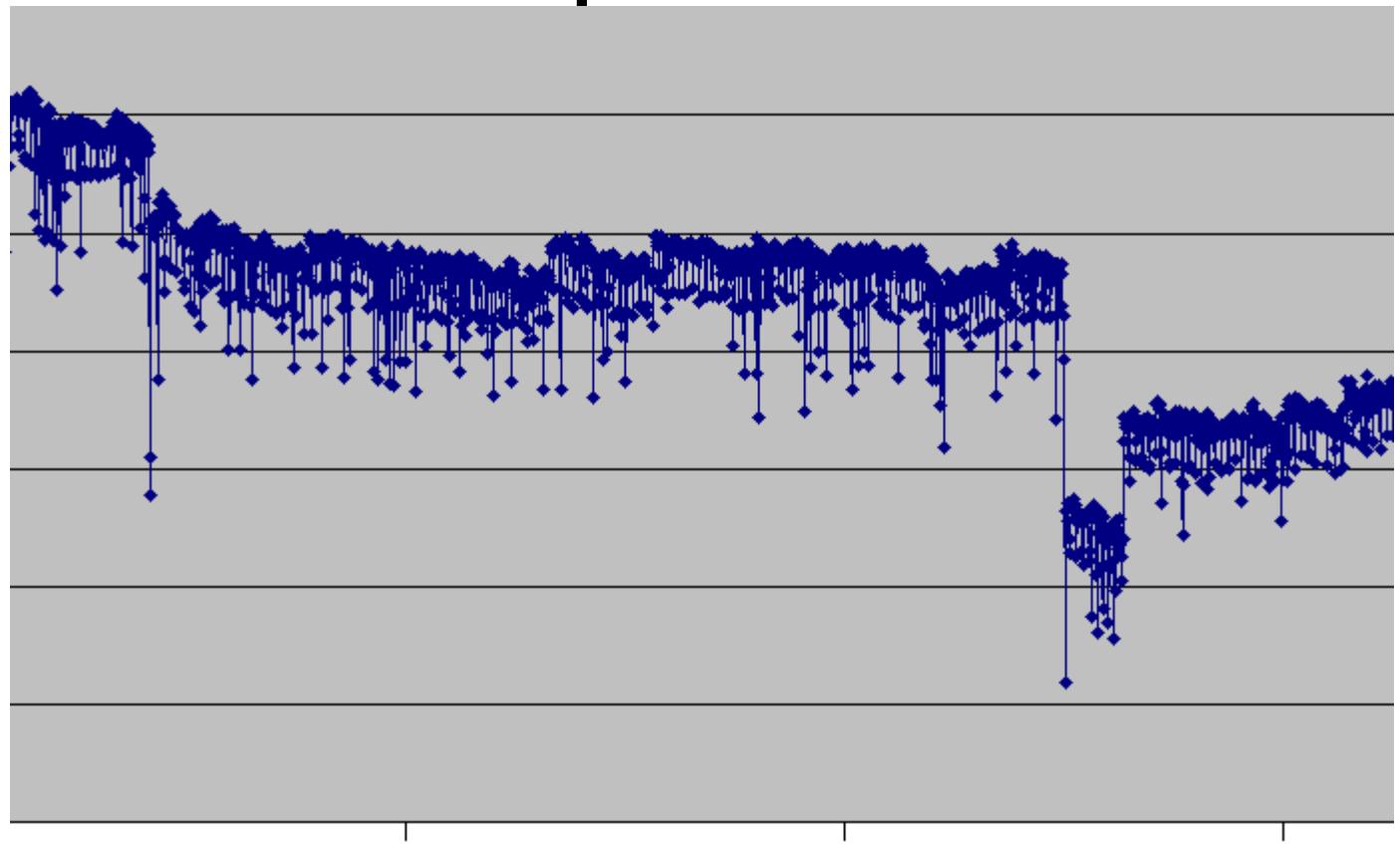
# Performance Testing Experiments

\\\OTT-SRV02\Memory\Available Bytes  
Single Continuous Session - One User -  
I:\scripts\AmiStr\CRS3g\AmiStr00A





# Performance Testing Experiments

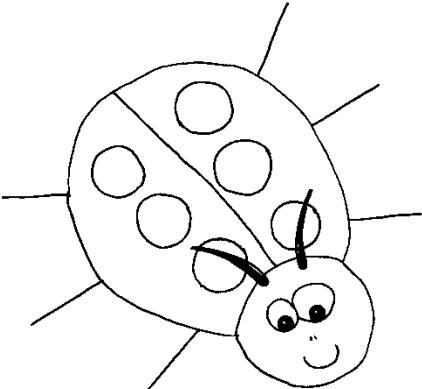


.0

24:00.0

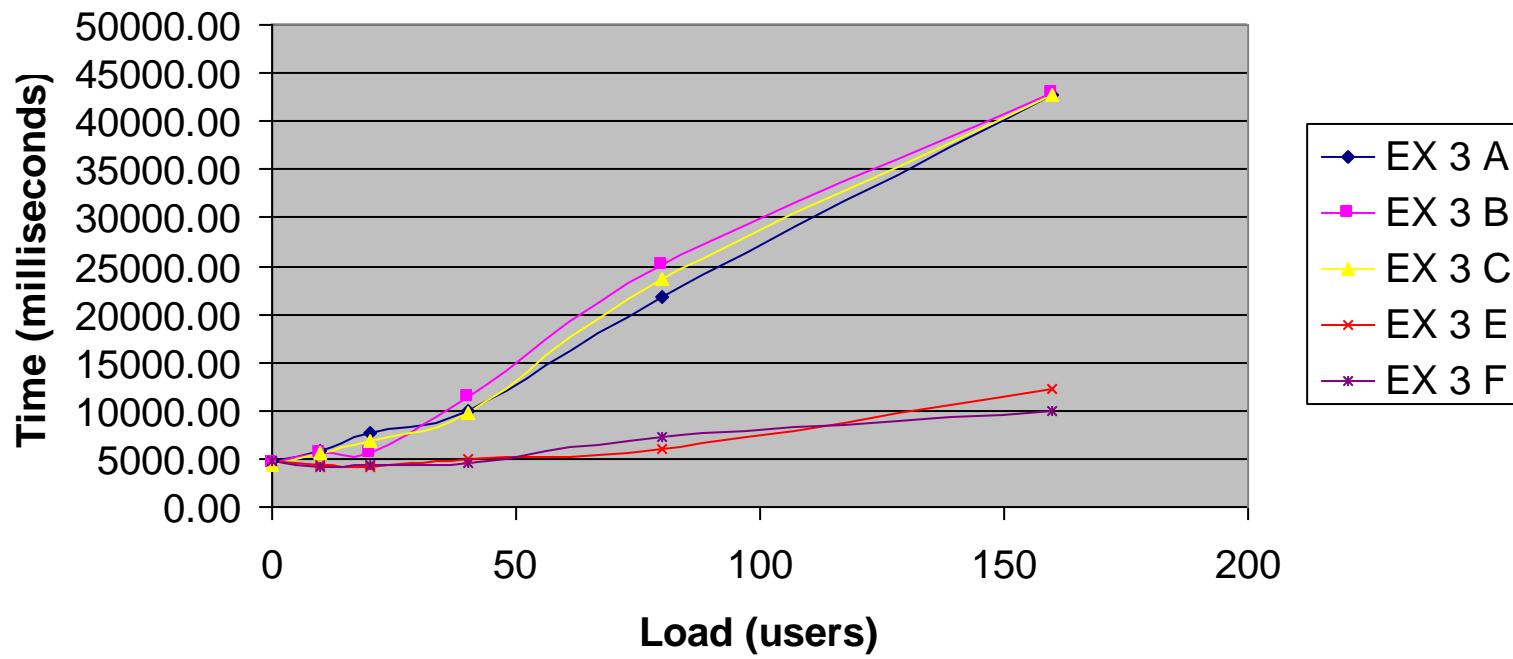
48:00.0

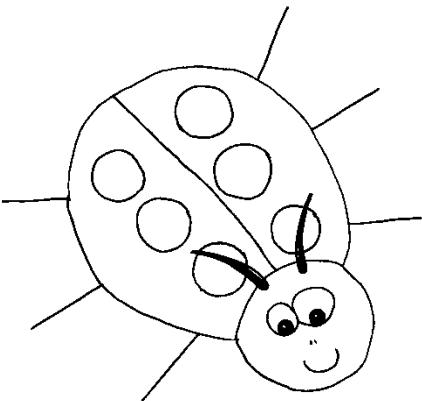
12:00.0



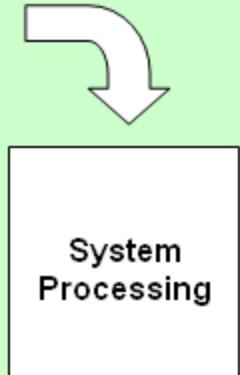
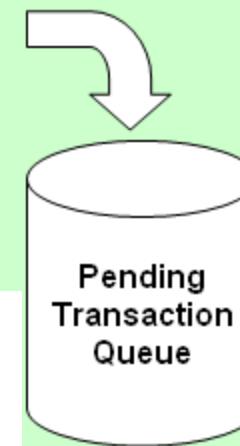
# Performance Testing Experiments

Transaction Time Stress Test Experiment 3

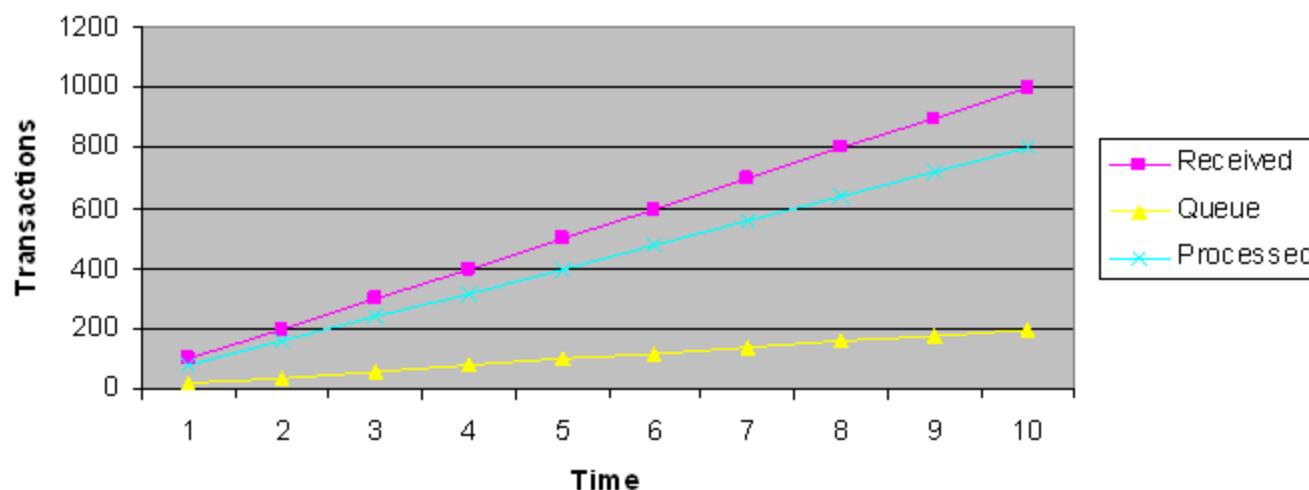


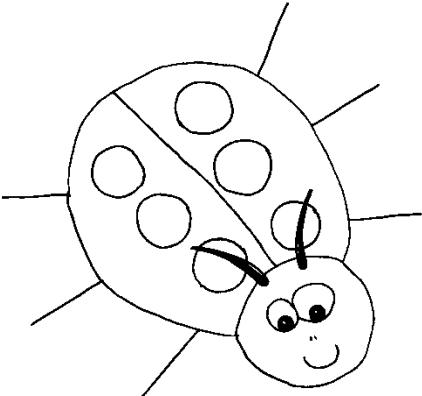


# Related Rates



**Linear Flow Rates**





# Minimal Platform

[Sign in](#) or create a [new account](#).  
[Cart](#) | [My Account](#) | [Track Order](#) | [Help Registry](#) | [Wish List](#) | [Gift Cards](#)

Apparel ▾ Baby ▾ Electronics ▾ Entertainment ▾ Home ▾ Jewelry ▾ Pharmacy ▾ Photo ▾ Sports ▾ Toys ▾ In Store Now ▾

SEARCH Electronics ▾ FOR  FIND See all departments

You are here: [Home Page](#) > [Electronics](#) > [Computers](#) > [Desktop Computers](#) > Desktop with Monitor

## ELECTRONICS : DESKTOP WITH MONITOR

**SPECIAL OFFERS**

[Clearance](#)  
[New](#)  
[Rollbacks](#)

**SHOP ELECTRONICS**

[Electronics](#)  
↳ [Computers](#)  
[Accessories & Peripherals](#)  
[Calculators](#)  
[Desktops](#)  
▪ [Desktops With Monitors](#)  
▪ [Desktops Without Monitors](#)  
▪ [Vista - Preloaded](#)

**Featured Item**

**eMachines W3623 Desktop PC w/ 17" LCD Monitor, Intel Pentium 4 processor 641 with Hyper-Threading technology**



- 17" widescreen LCD display
- 3.2 GHz Intel Pentium 4 processor 641
- 512 MB RAM, 160 GB hard drive
- DVD Burner, Windows Vista Home Basic

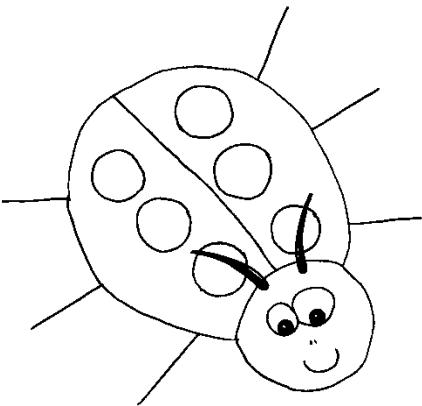
**\$498.00**  
Was: \$598.00

[ADD TO CART ▶](#)

Items 1-11 of 11 total      Sort by: [Featured Items ▾](#)

Narrow your results by:

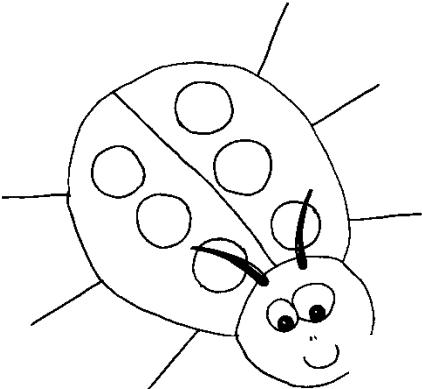
Brand ▾ Price ▾ Processor Type ▾ Monitor Type ▾  
Processor Brand ▾ Delivery Option... ▾ Special Offers ▾ Customer Rating ▾



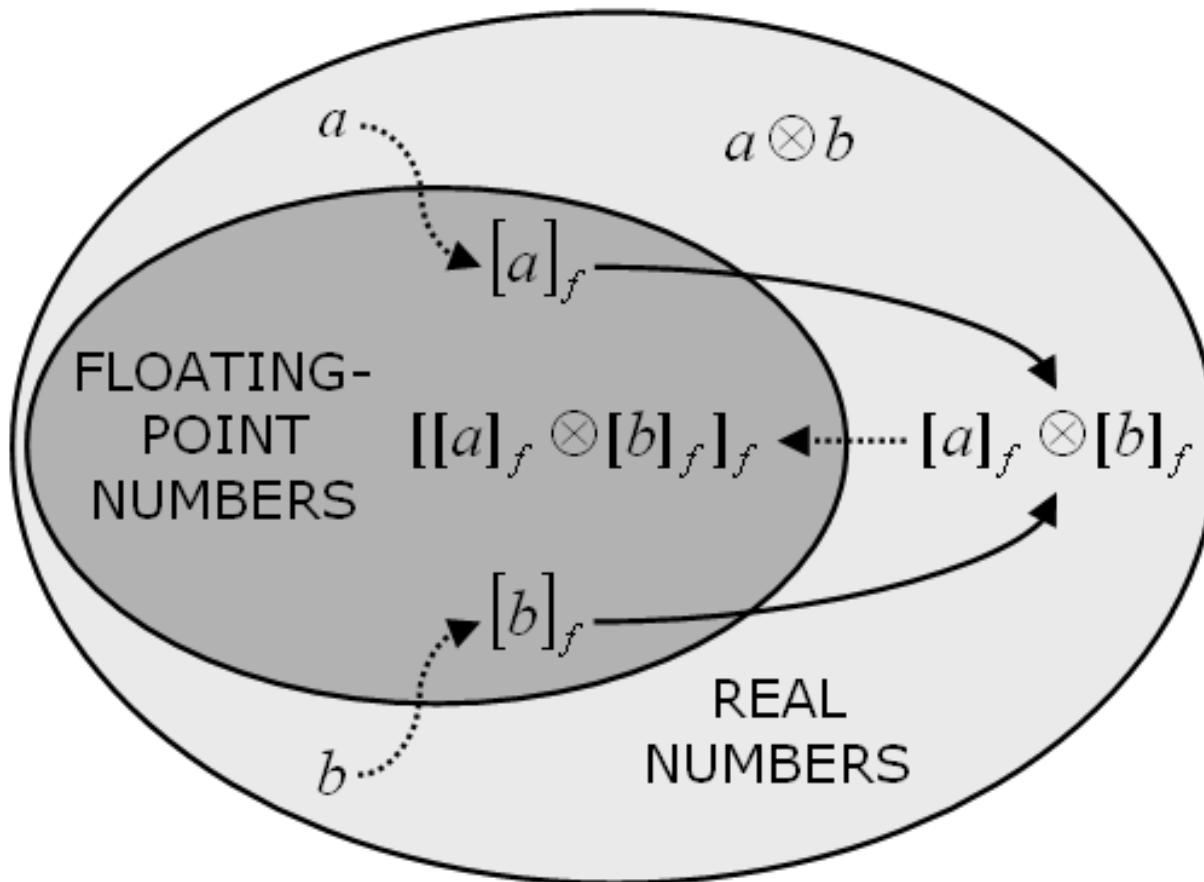
# Variable Type Boundaries

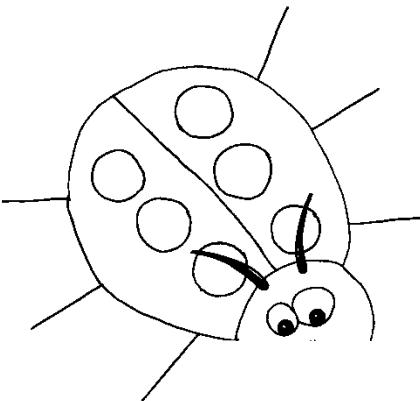
- Numbers and Calculations
  - Computer Representations
    - Integer
    - Floating Point
    - Casting
  - Precision
    - Epsilon
    - Rounding
    - Error

unsigned char	0	255
short int	-32,767	32,767
unsigned short	0	65,535
int	-32,767	32,767
unsigned int	0	65,535
long int	-2,147,483,647	2,147,483,647
unsigned long	0	4,294,967,295



# Variable Type Boundaries

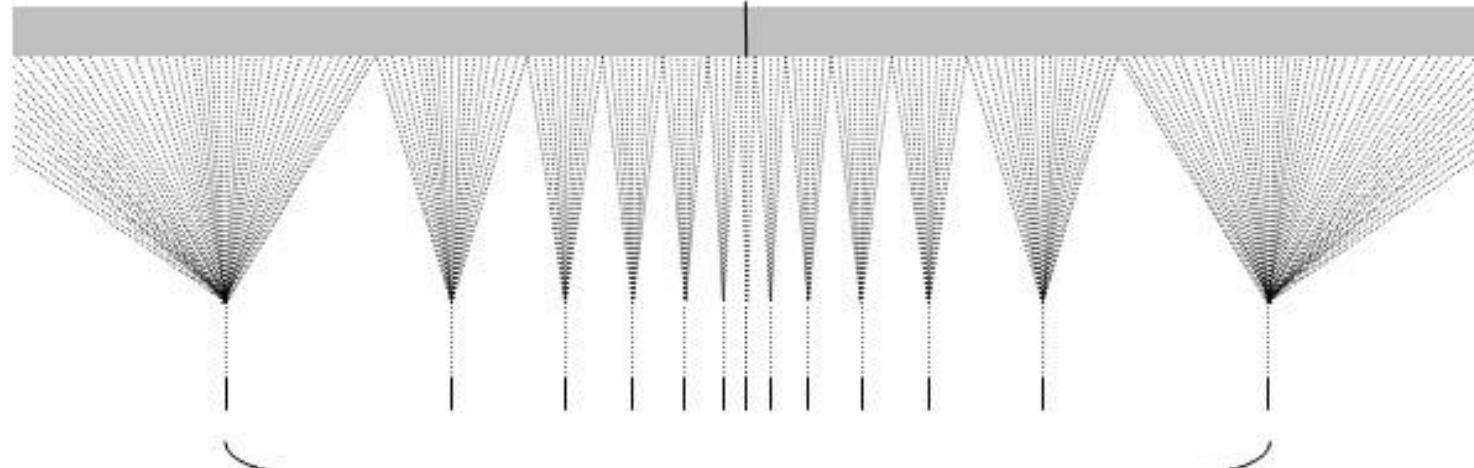




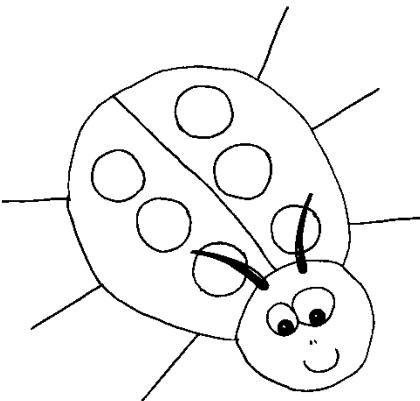
# Variable Type Boundaries

REAL NUMBERS

0



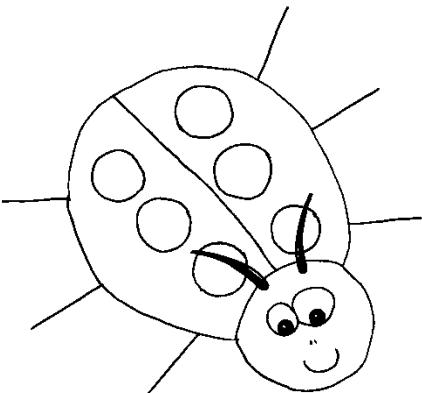
FLOATING-POINT  
NUMBERS



# Variable Type Boundaries

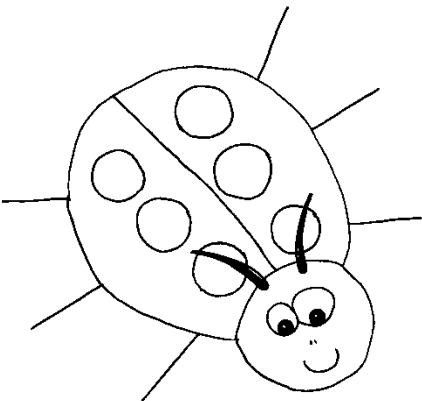
- Times
- Dates
- Causality
- Ranges
- Formats



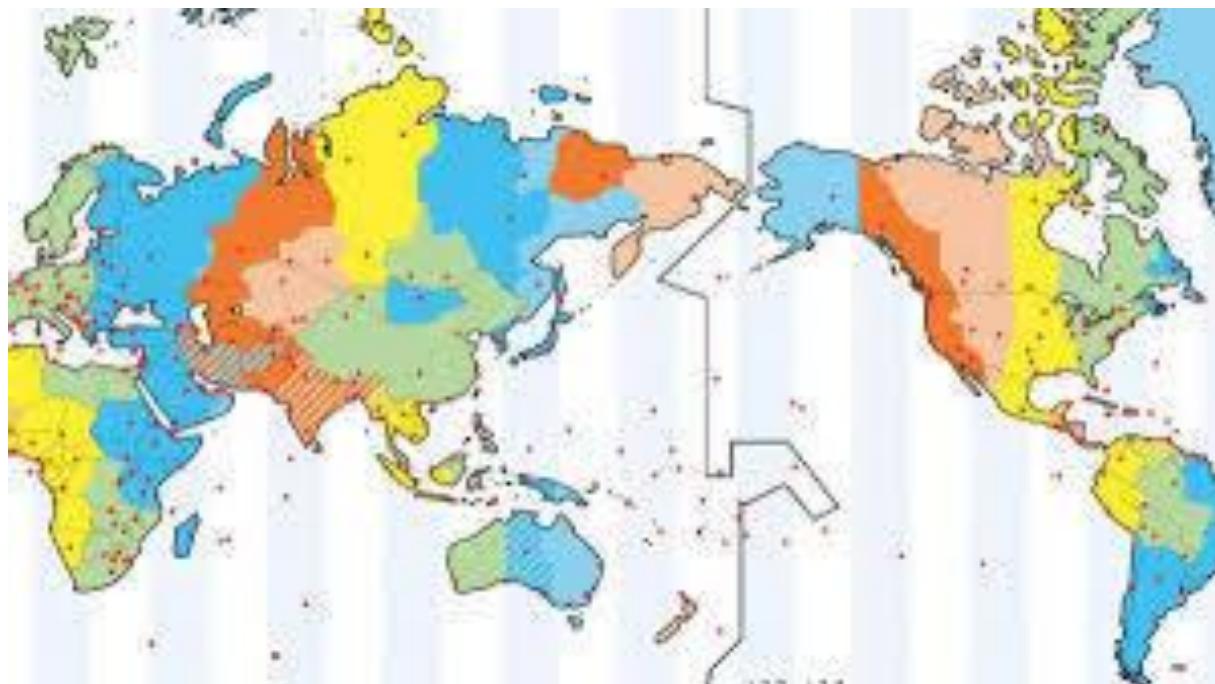


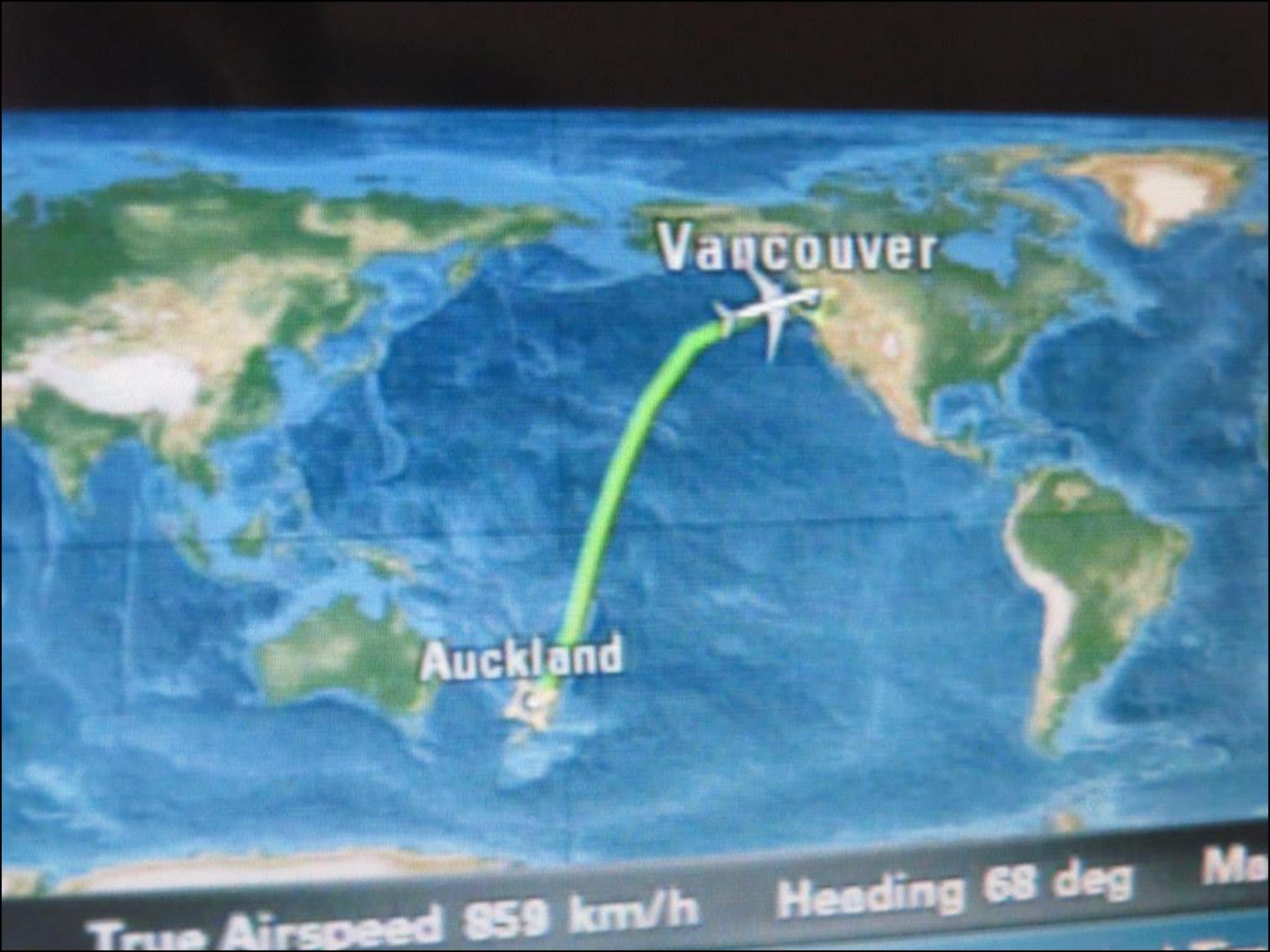
# Date/Time Boundaries

Day	First Day of Month	Last Day of Month
Month	First Month of Year	Last Month of Year
Year	First Year of Century	Last Year of Century
Year	First Year of Millennium	Last Year of Millennium
Year	Leap Year	
Year	Leap Century	
Hour	First hour of day	Last hour of day
Hour	First hour of morning	Last hour of morning
Hour	First hour of afternoon	Last hour of afternoon
Hour	Leap hour forward	
Hour	Leap hour backward	
Day	Leap day forward	
Day	Leap day backward	
Minute	First minute of hour	Last minute of hour
Second	First second of minute	Last second of minute



# Date/Time Boundaries





Vancouver

Auckland

True Airspeed 859 km/h

Heading 68 deg

Mo

## Google

Gmail



+Robert



Share



More

1-100 of 29,379



COMPOSE

Primary

Social

Promotions



Inbox

Important

Sent Mail

Circles



Search people...

- Duncan Nisbet
- Marlena Compton on Me+ (<http://m.me/rebsab>)
- Pradeep Soundar... Pregnancy Analo...
- Raymond Chan
- ✉ Jonathan Li On W... ([http://www.hanafonline...](http://www.hanafonline.com))
- ✉ Adam White

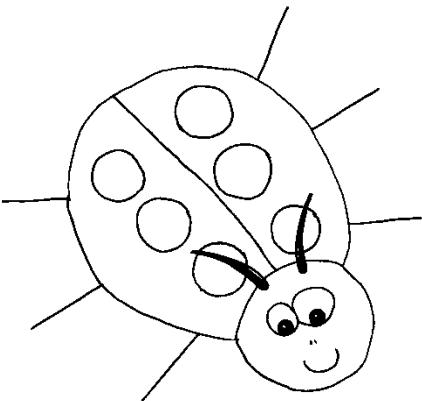


<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> Mastura, Robert (4)	FW: SOFTEC2014 - Hi Mastura, Thank you very much indeed. I confir...	8:17 pm
<input type="checkbox"/> <span style="color: purple;">✉</span> <span style="color: orange;">✉</span> mobility confirmation	Your e-bill is ready - Bell Log in to MyBell   Register now Hello , Your F...	7:01 pm
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> Julie, Robert (7)	May 8-9 - Thanks. Let me know. Julia MacNaughton Senior Training S...	3:59 pm
<input type="checkbox"/> <span style="color: purple;">✉</span> <span style="color: orange;">✉</span> Akaash, Robert (4)	Jonar Talk - Hi Rob, Hope all's well. I just wanted touch base to see if y...	11:33 am
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> pict (10)	PICT - See Attached.	10:33 am
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> Skype (2)	You have a new voice message - This is an automated email, please d...	9:26 am
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> Amazon.ca Customer Servi...	Your savings from Amazon.ca (order #701-1404761-1145047) - (Vous...	9:12 am
<input type="checkbox"/> <span style="color: purple;">✉</span> <span style="color: orange;">✉</span> Bell Canada	Your e-bill is ready - Your e-bill is ready Account summary Account: 19...	5:57 am
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> James Lyndsay (5)	[LEWT general] Next LEWT: Sunday 2 March - Reply ABOVE THIS LINE...	2:25 am
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> James Lyndsay (2)	Re: [LEWT general] LEWT! Next LEWT? - Reply ABOVE THIS LINE to...	12:57 am
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> Robert via Dropbox (2)	Robert Sabourin shared "Mobile_Testing" with you - From Robert: "Mol...	Feb 19
<input type="checkbox"/> <span style="color: yellow;">★</span> <span style="color: orange;">✉</span> Lalith, Robert (3)	JIT - Workshop at Wellington on 17/02/14 - Hi Robert, Just aline to tha...	Feb 19
<input type="checkbox"/> <span style="color: orange;">✉</span> Software Conferen...	Brochure Now Available for Download - Download the Brochure! Havin...	Feb 19
<input type="checkbox"/> <span style="color: orange;">✉</span> Robert, Robert (2)	Re: EXAM on Mon, April 7 - Foundation & Advanced - I've copied Rob...	Feb 19
<input type="checkbox"/> <span style="color: orange;">✉</span> Robert, Robert (4)	Speaking engagement/Talk - Hey Rob, I had a potential client reach ou...	Feb 19
<input type="checkbox"/> <span style="color: orange;">✉</span> Robert, Michael (2)	AA Progressive Visit - Hi Greg I hope all is well - and I apologize for the...	Feb 19
<input type="checkbox"/> <span style="color: orange;">✉</span> Robert, Lee (3)	In NZ - Hi Lee Got your message - I am in NZ - back in Montreal on Fri...	Feb 19
<input type="checkbox"/> <span style="color: orange;">✉</span> Robert Sabourin	Message from Ice CopeLand - Rob call back lee. 18003366644 Messa...	Feb 19

100%

8:41 PM

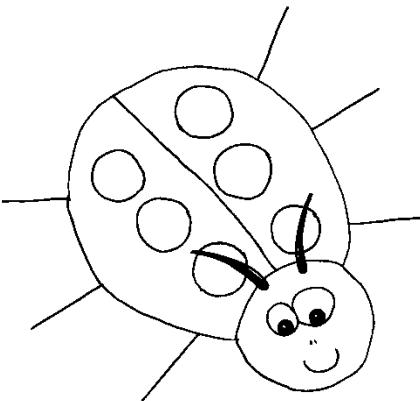
20/02/2014



# Date/Time Boundaries

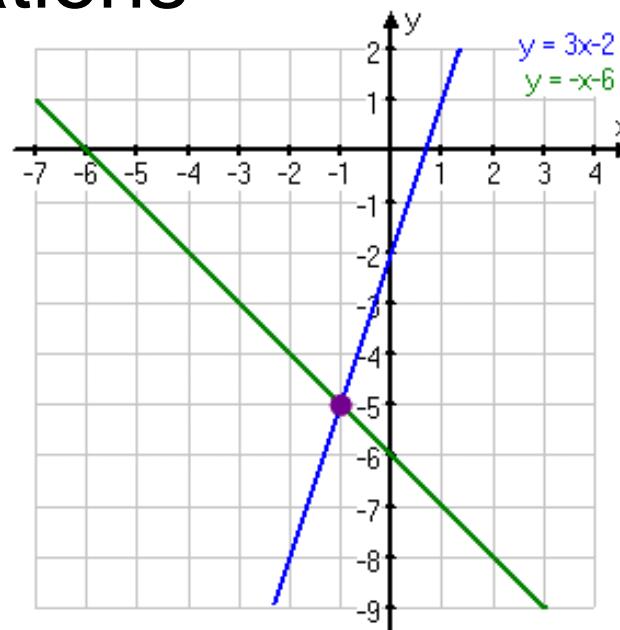
## Transactions Crossing Time Ranges

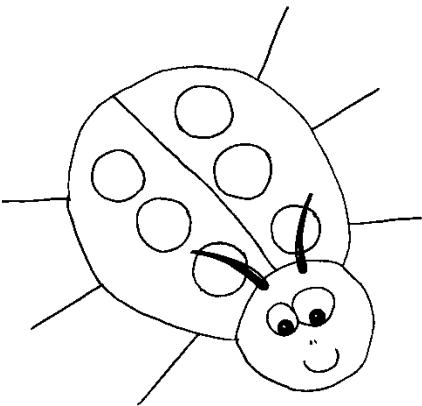
- Crossing hour (for example Newfoundland time zone is off by 30 minutes from Atlantic time zone)
- Crossing AM PM (in one time zone transaction is in morning, in another time zone transaction is in the afternoon)
- Crossing Day (in one time zone transaction is in different day)
- Crossing Year (in one time zone transaction is in different year)
- Crossing Month (in one time zone transaction is in different month)
- Crossing Century (in one time zone transaction is in different century)
- Crossing Millennium (in one time zone transaction is in different millennium)



# Variable Type Boundaries

- Systems of equations
  - Matrices
  - Condition

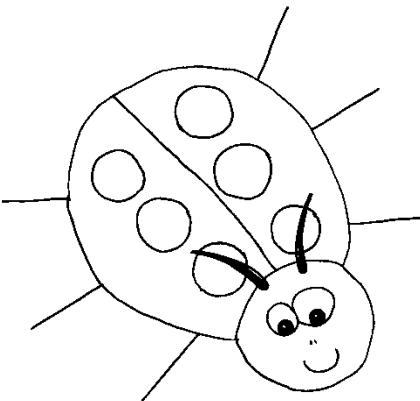




# Variable Type Boundaries

- Text
  - Strings
  - Collation
  - Syntax





# Bidirectional Text Boundaries

Arabic-Indic digits with non-breaking hyphen

٣١-١٢-١٩٩٩

١٩٩٩-١٢-٣١

Arabic-Indic digits with slash

م ٣١/١٢/١٩٩٩

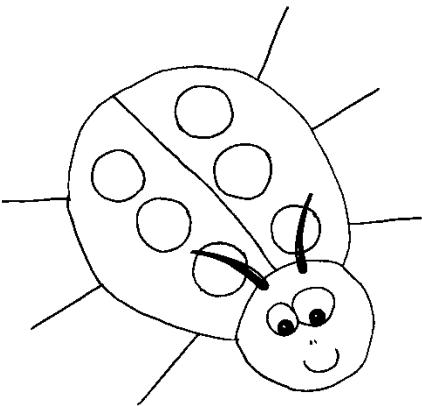
م ١٩٩٩/١٢/٣١

Letters and digits

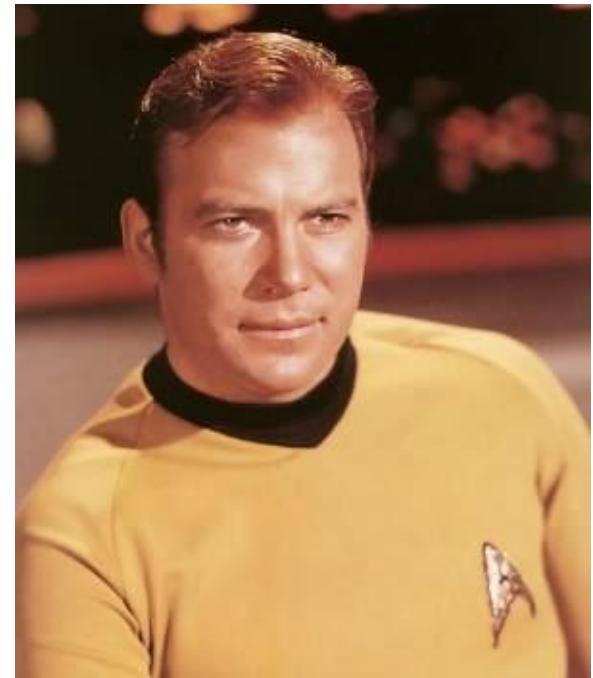
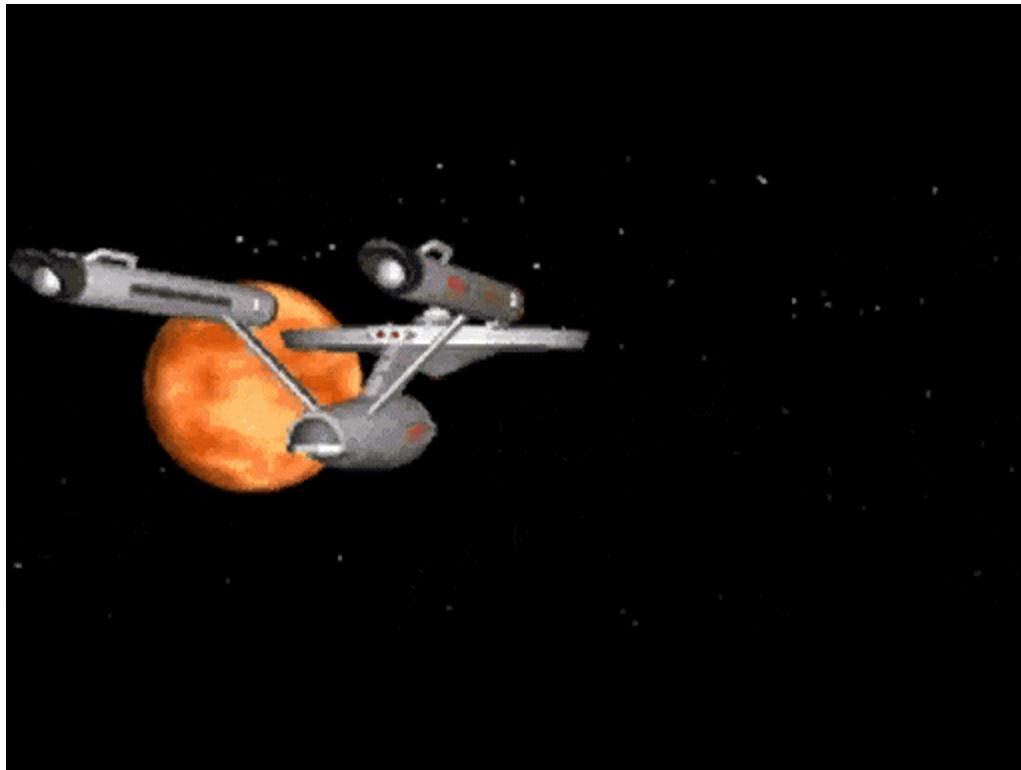
© 2004 Google – 90 00 000 ويب صفحات کی تلاش ہو رہی ہے

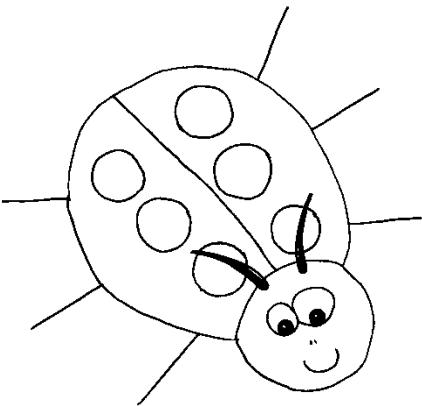
© 2004 Google – 90 00 000 ويب صفحات کی تلاش ہو رہی ہے

© 2004 Google – 9000000 ويب صفحات کی تلاش ہو رہی ہے



# To Boldly Go

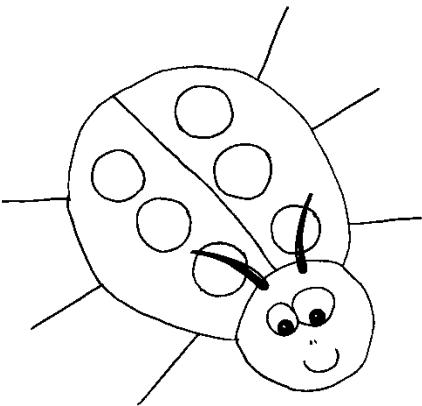




# Boundary

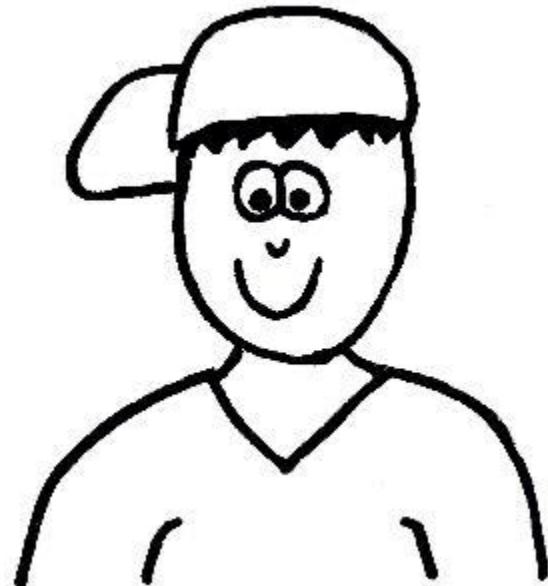
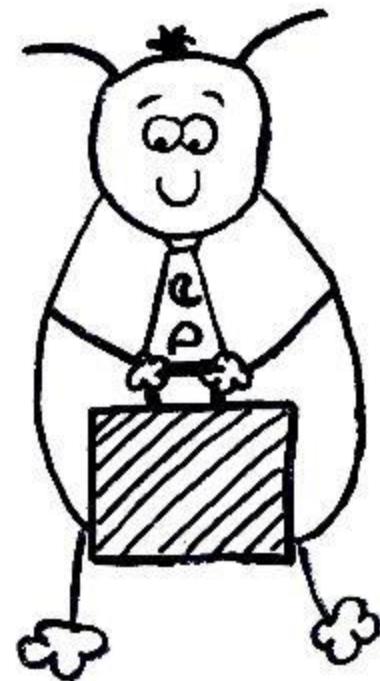
- Buzz Light-year
- To infinity and beyond

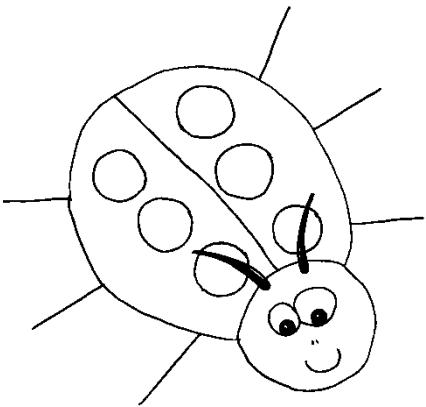




# Thank You

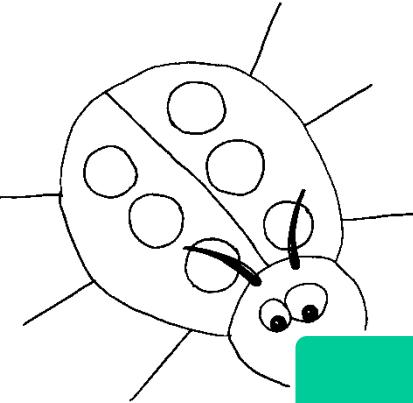
- Questions?





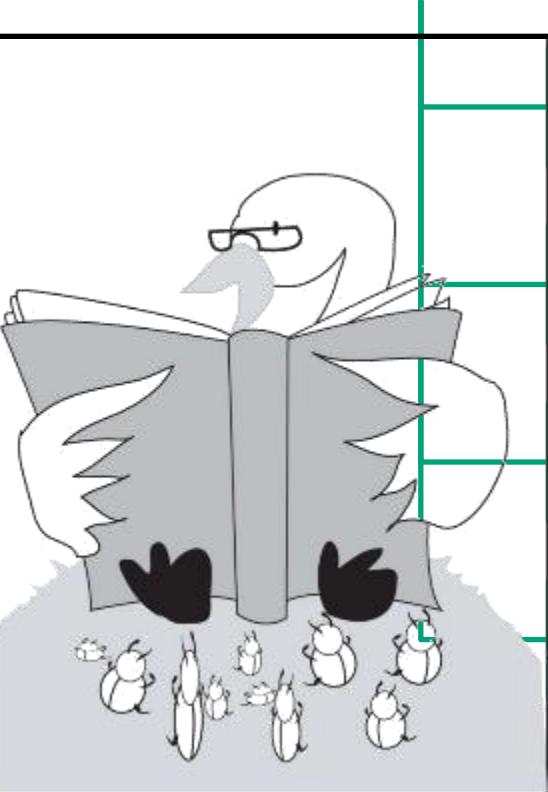
# Test Design

*Agile Story Acceptance Tests*



# Story Testing

## Story Tests

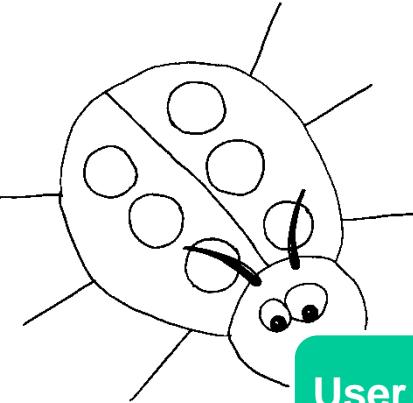


Elicited from customer

Clear examples

Confirm implementation

Demonstrate correctness

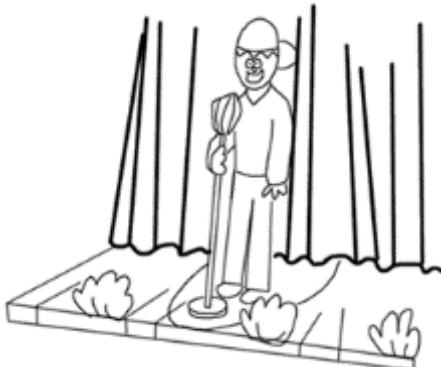


# Story Testing

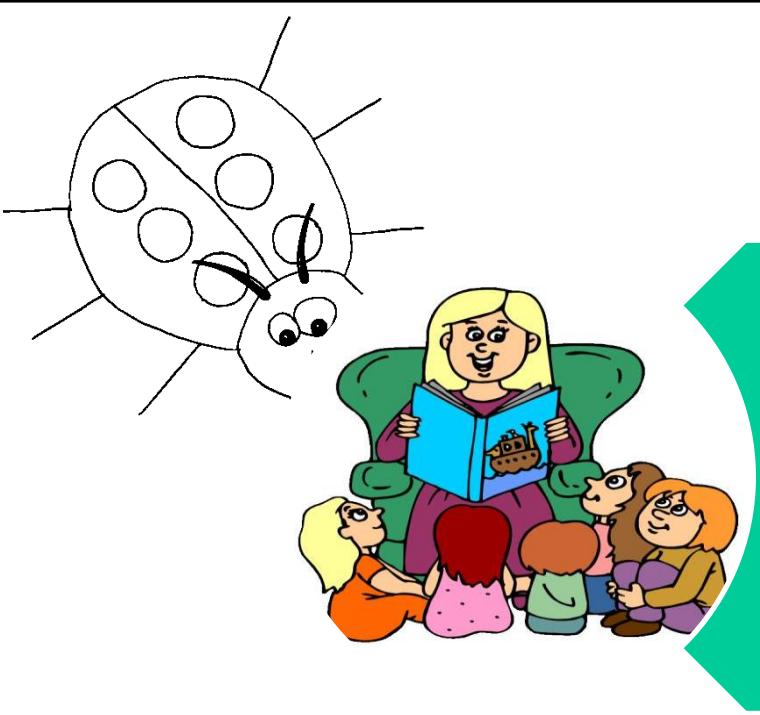
User stories are written by the customer and describe, in two or three sentences, what the system needs to do for them

The purposes of user stories is to:

- Provide a basis for development time estimates
- Replace large, formal requirements documents
- Drive the creation of automated acceptance tests



When the time comes to implement the story, developers will go to the customer to get the details face-to-face

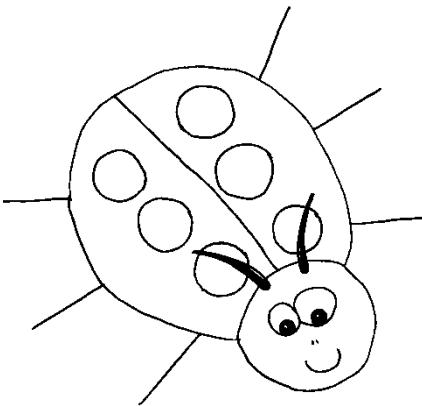


A User Story answers these questions:

- Who is the user?
- What do they want to do?
- For what benefit?



User Story Tests provide examples of different scenarios to confirm understanding of typical, alternate and error situations a user may encounter



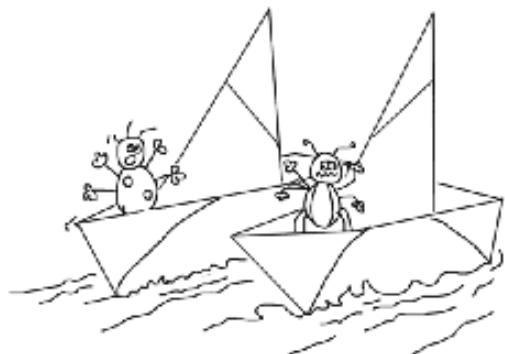
# Story Testing

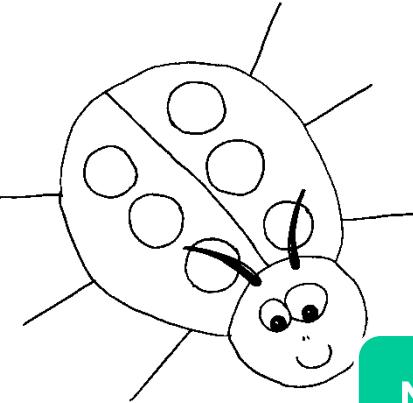
User  
Stories

Normal  
Scenarios

Alternative  
Scenarios

Error  
Scenarios

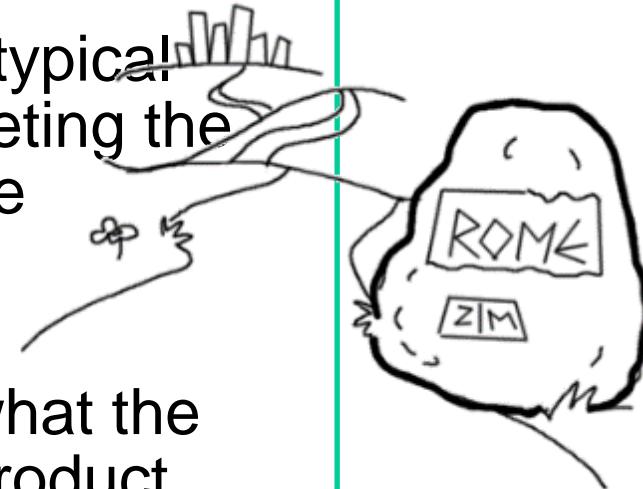


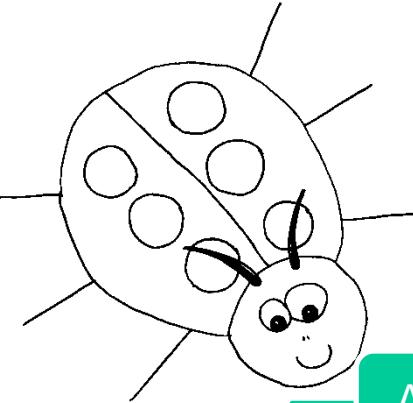


# Story Testing

## Normal Flows

- Define scenarios which are typical examples of the user completing the story with a positive outcome
- A normal flow is a clear representative example of what the customer really needs the product to do

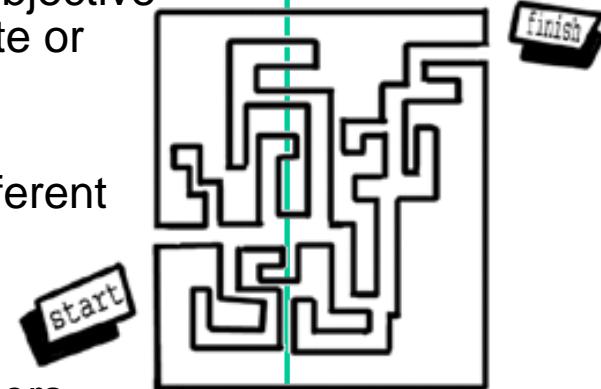


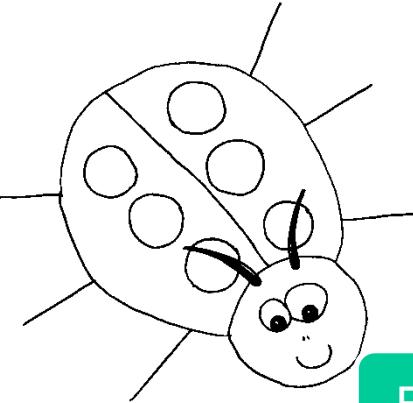


# Story Testing

## Alternate Flows

- Define scenarios which are examples of the user completing the story with a positive outcome but with a variation in the path to achieve the objective due to differences in user date, system state or other factors
- Alternative flows help explore the many different ways a user can accomplish the story
- Elaborating alternative flows improves implementation estimates and clears up users understanding of the user stories scope

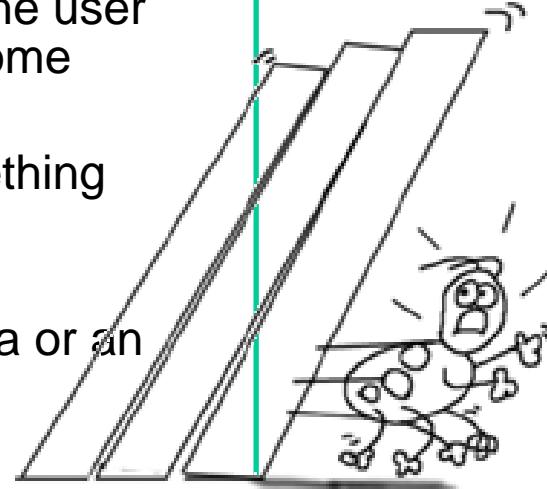


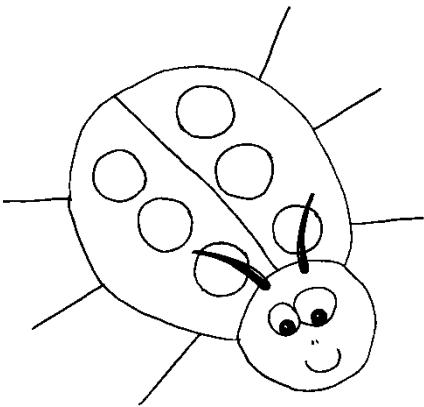


# Story Testing

## Error Flows

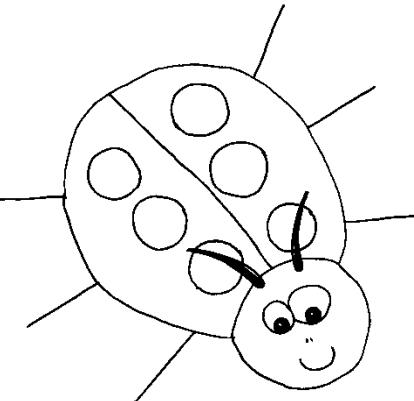
- Define scenarios which are examples of the user completing the story with a negative outcome
- These are alternative flows in which something goes has gone wrong
- Error flows can be triggered by invalid data or an inappropriate system state
- Eliciting error flows helps the implementers understand the type of error handling expected



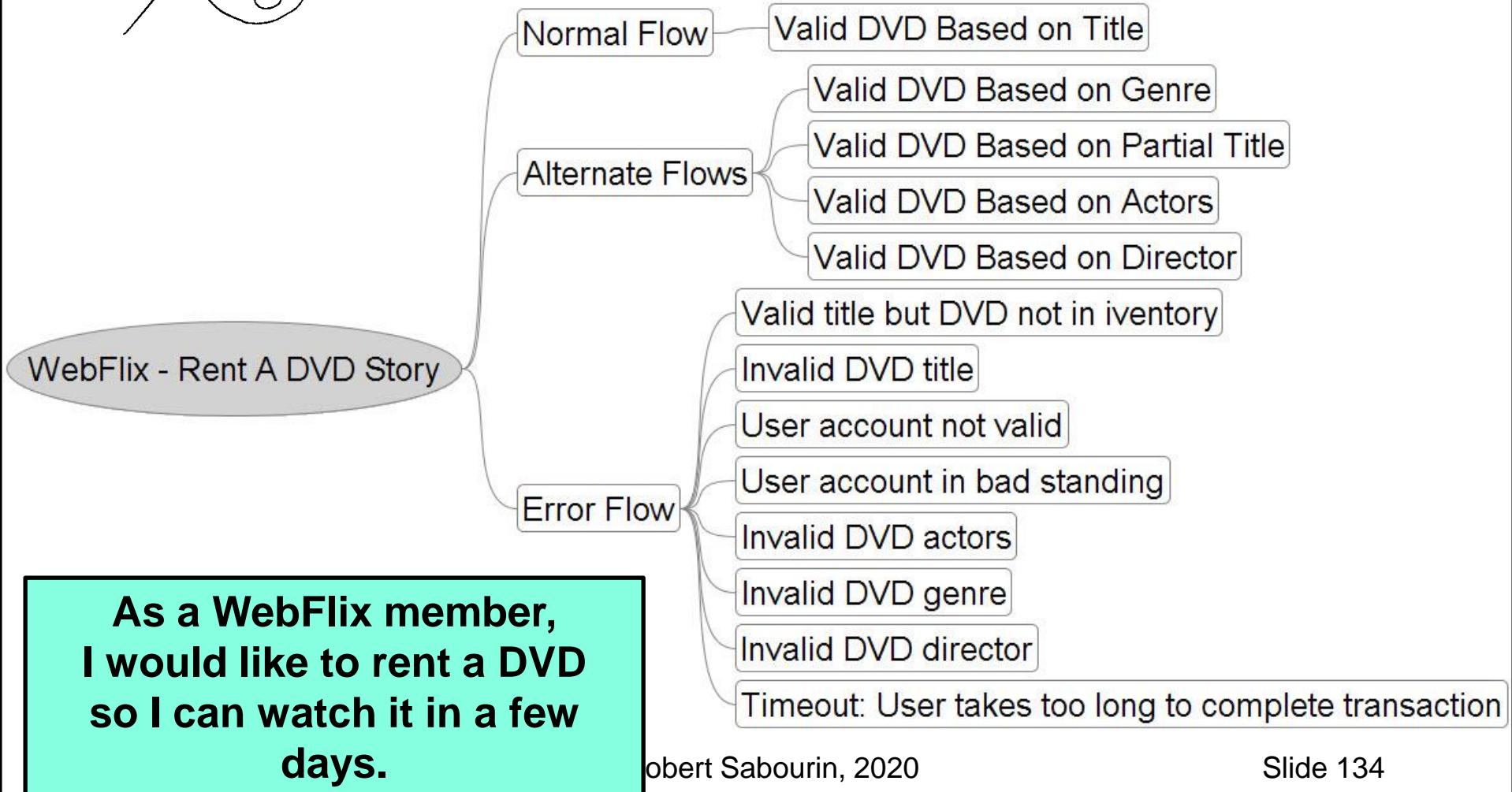


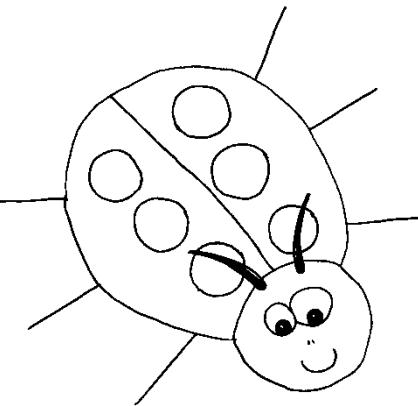
# WebFlix DVD Rental

*Story Testing*

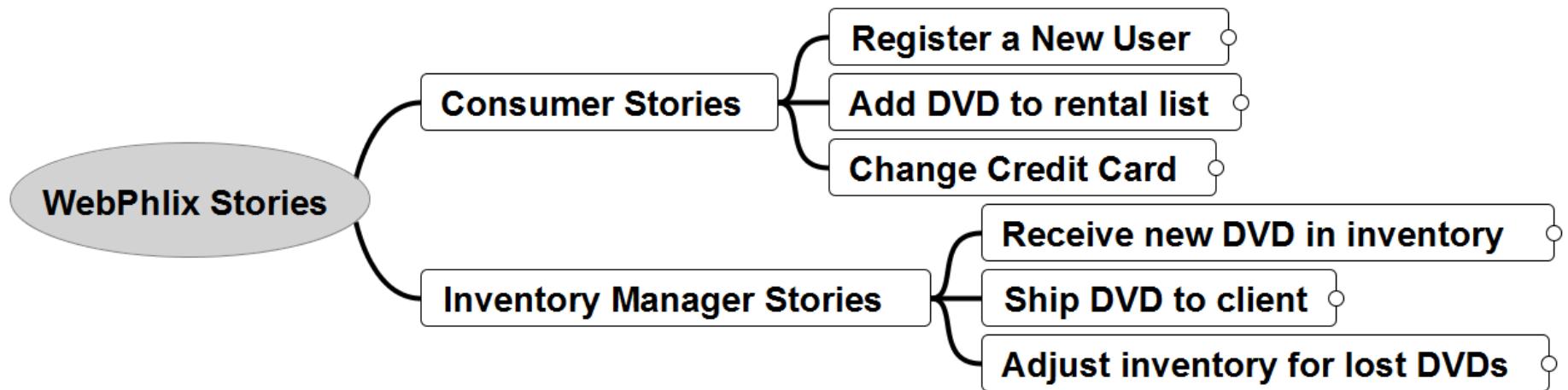


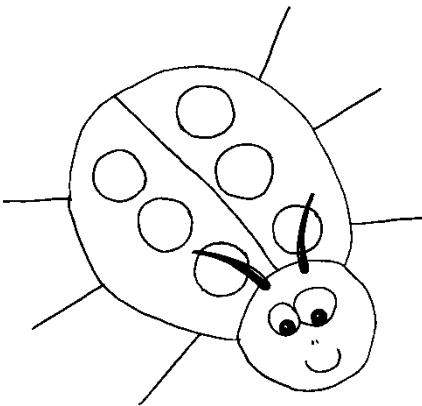
# Story Testing



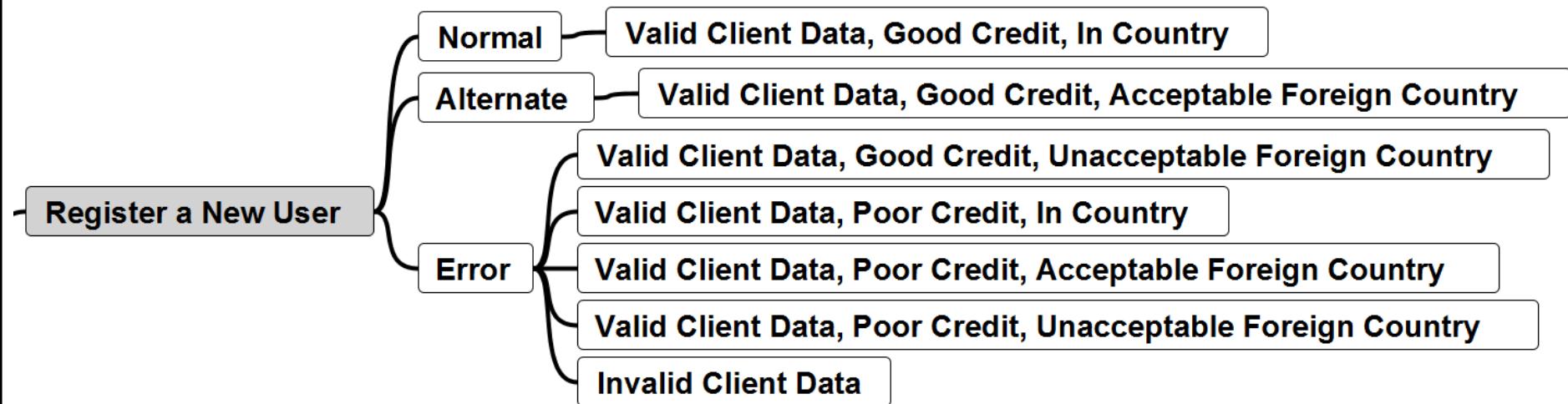


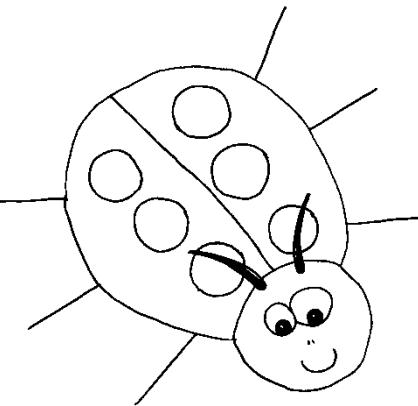
# Story Testing



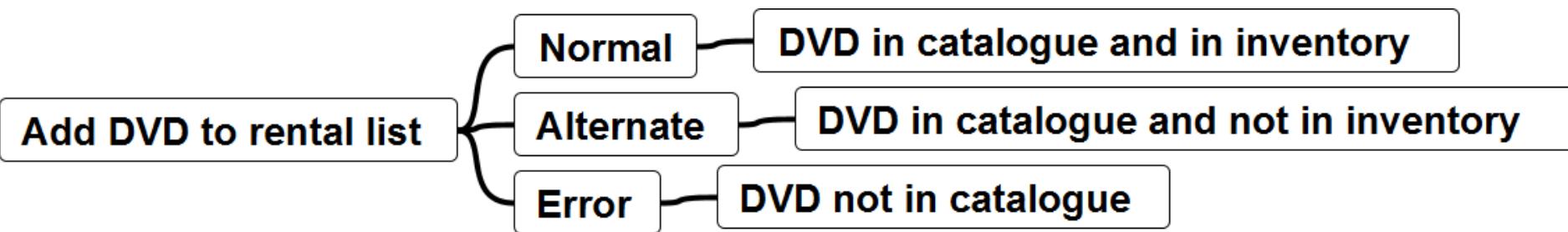


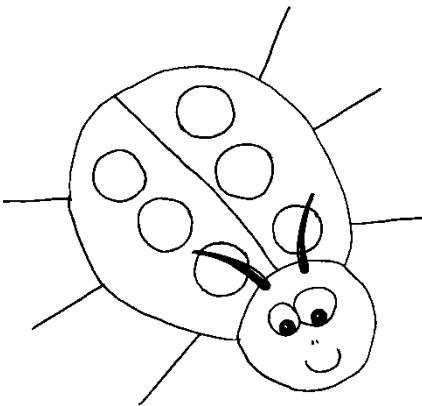
# Story Testing



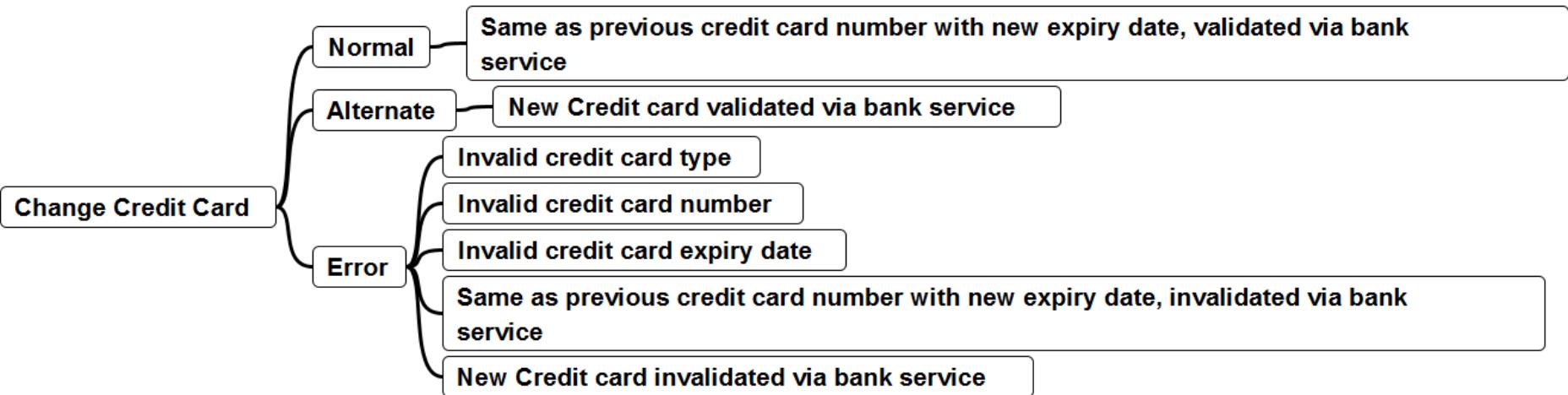


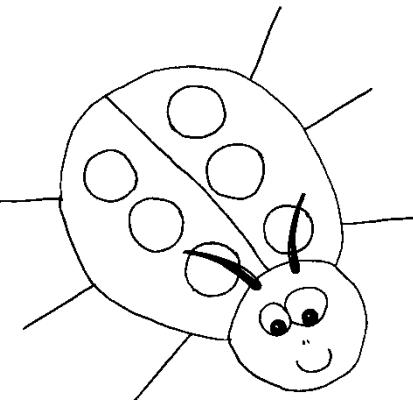
# Story Testing



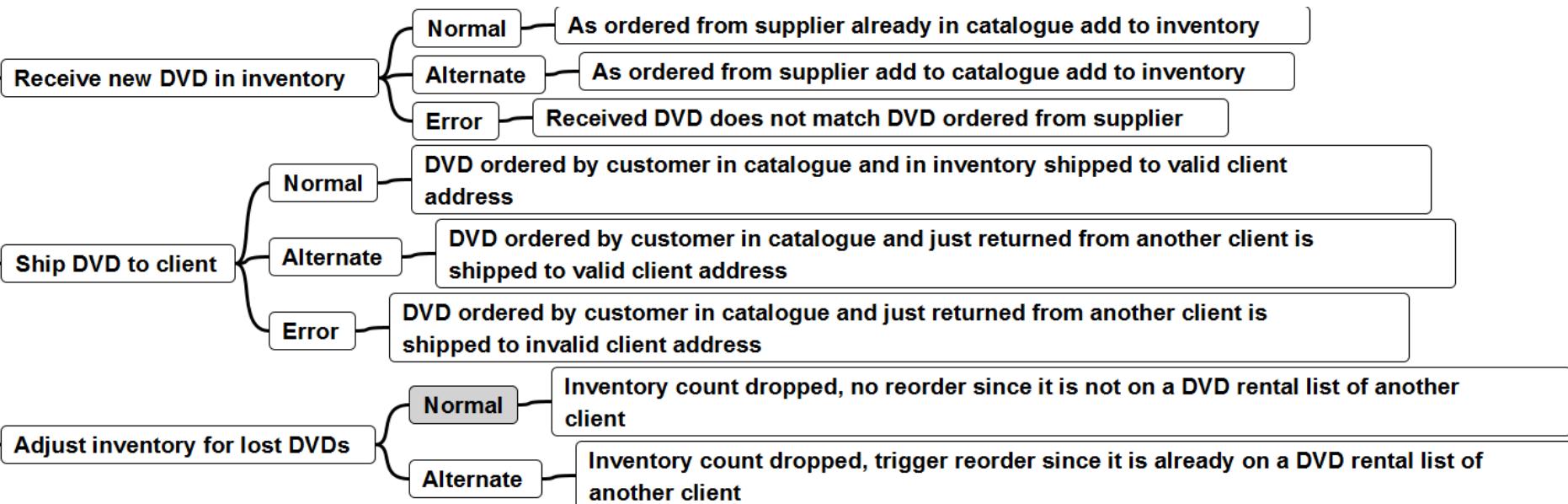


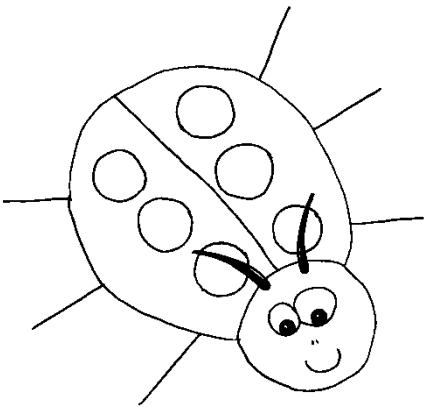
# Story Testing





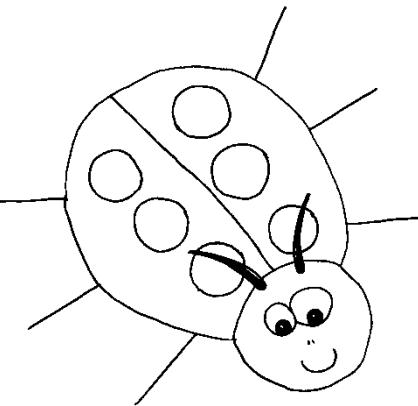
# Story Testing



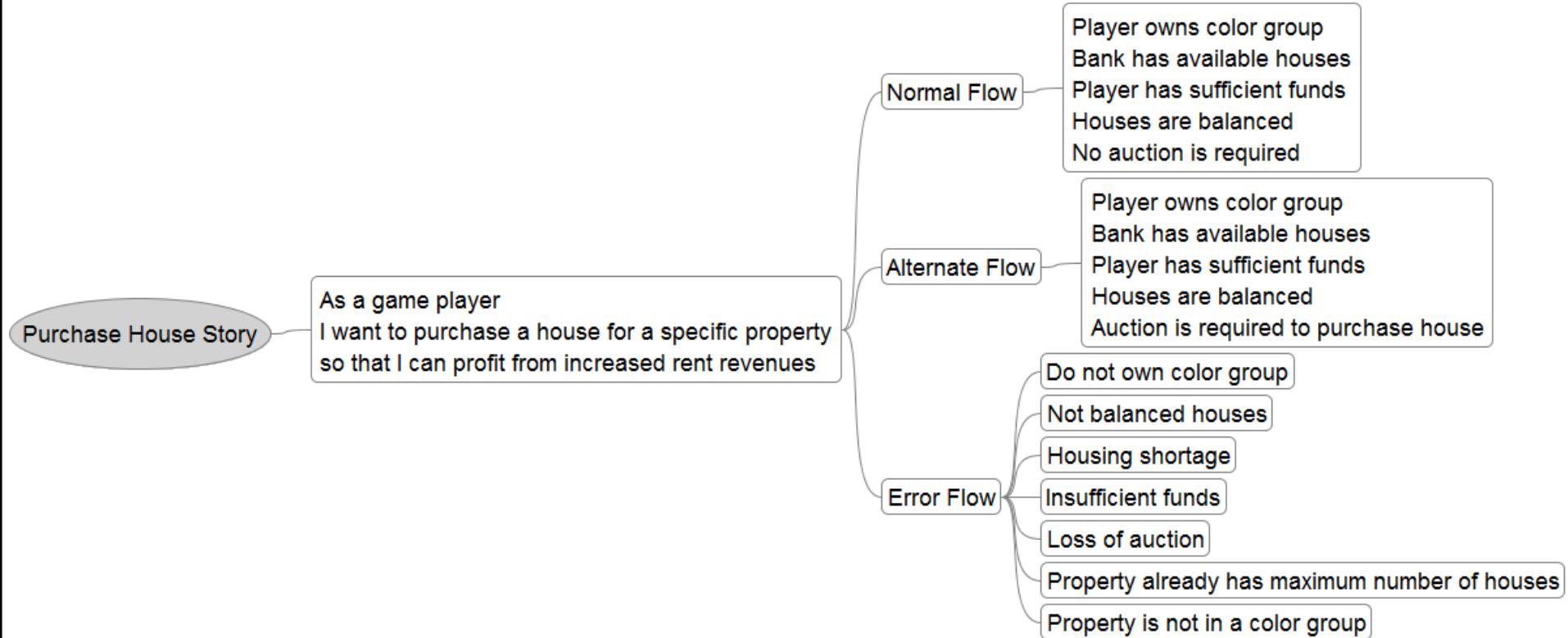


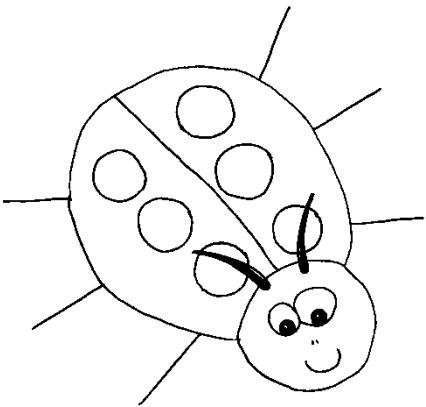
# Monopoly® Game

*Story Testing*



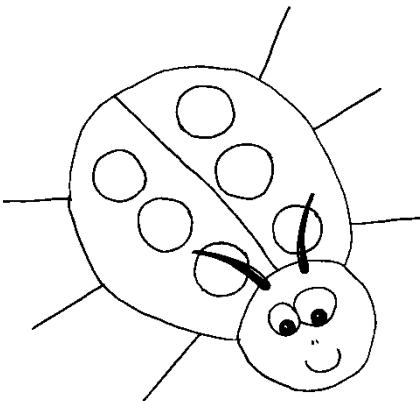
# Story Testing





# Test Design

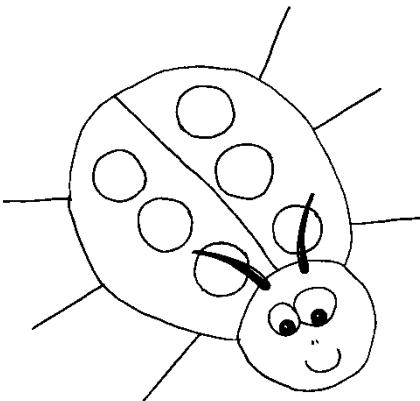
*Usage Scenario Tests*



# Uncovering User Needs

- John Kennedy's inaugural address  
January 20, 1961
- **“...ask not what your country can do  
for you - ask what you can do for  
your country...”**





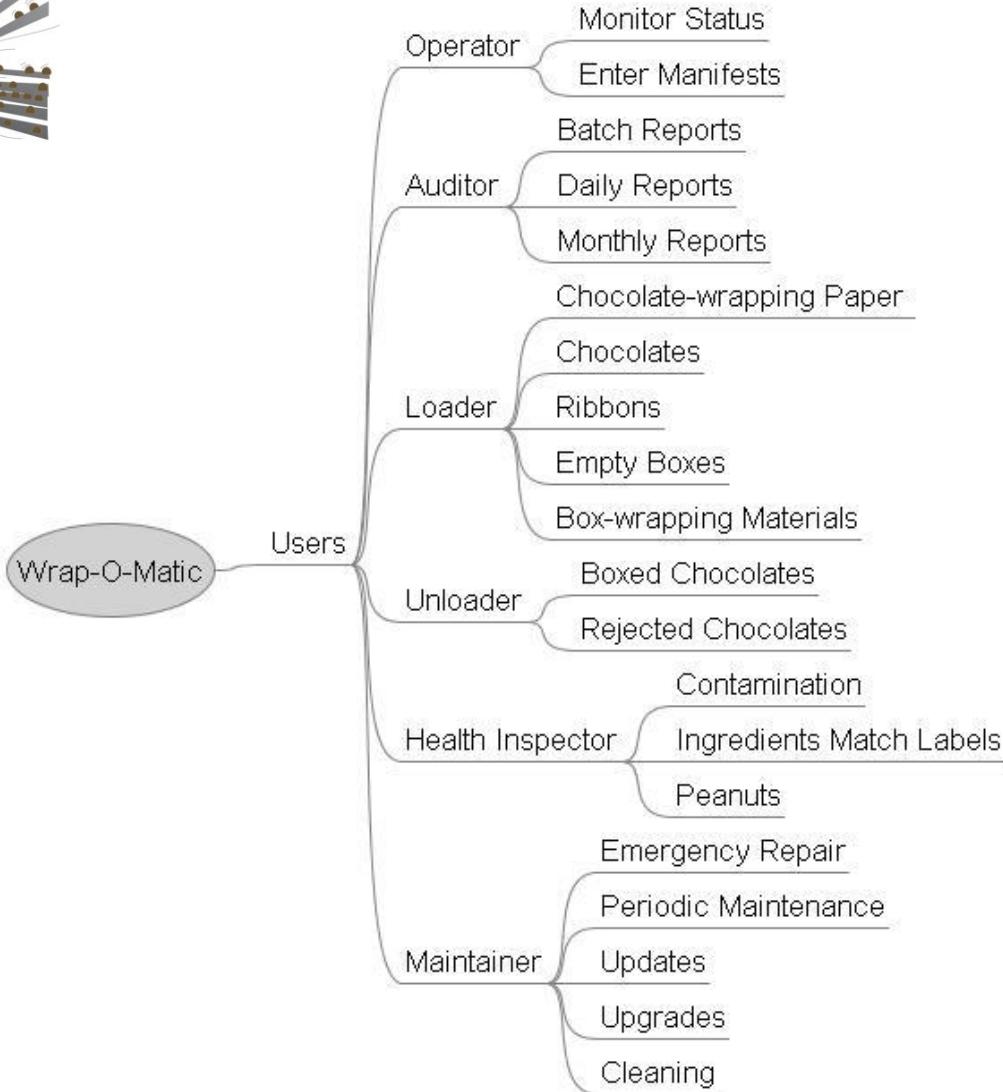
# Uncovering User Needs

- “...ask not what your system can do for the user - ask what your user does with the system...”

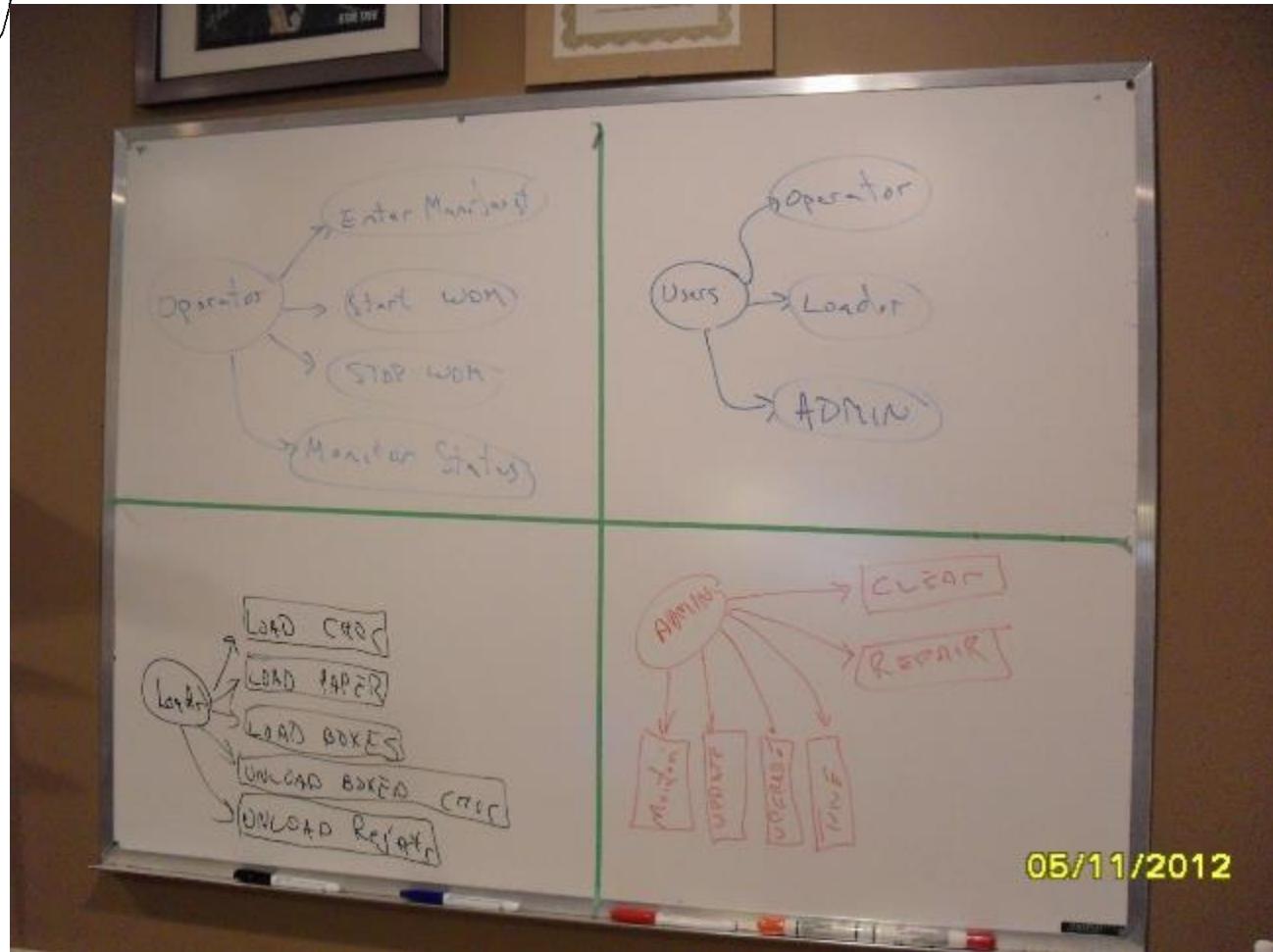


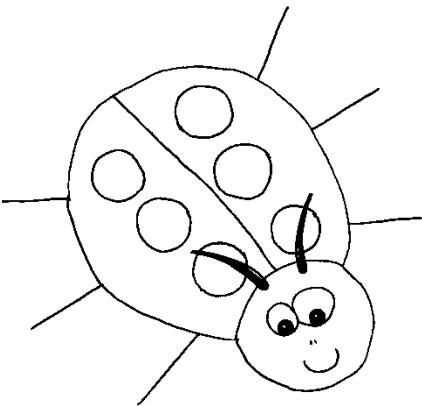


# Scenario Based Testing



# User Mind Map

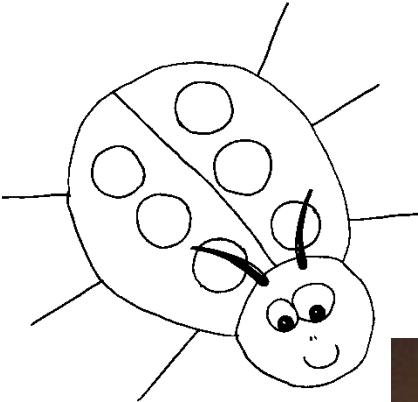




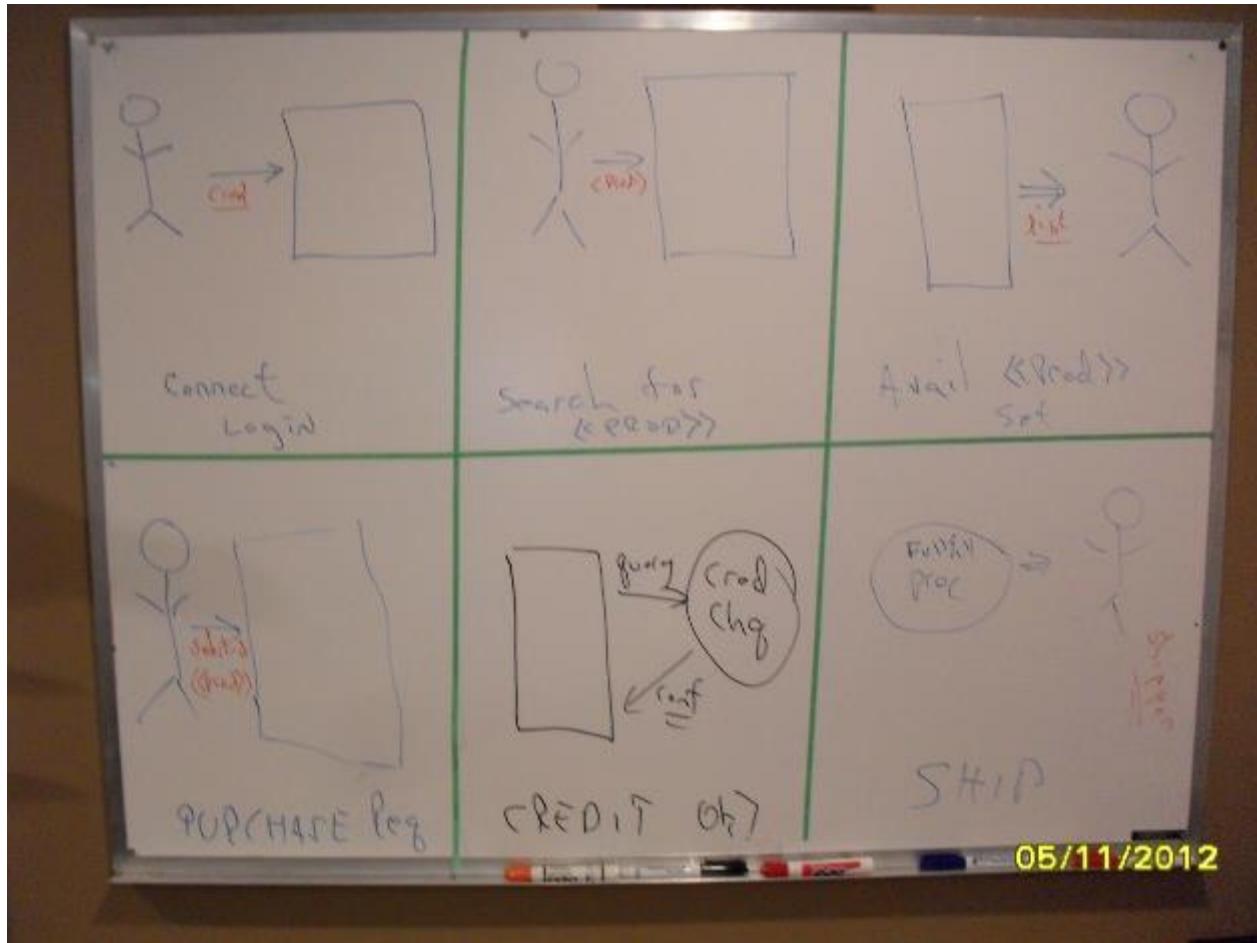
# Buying a Book

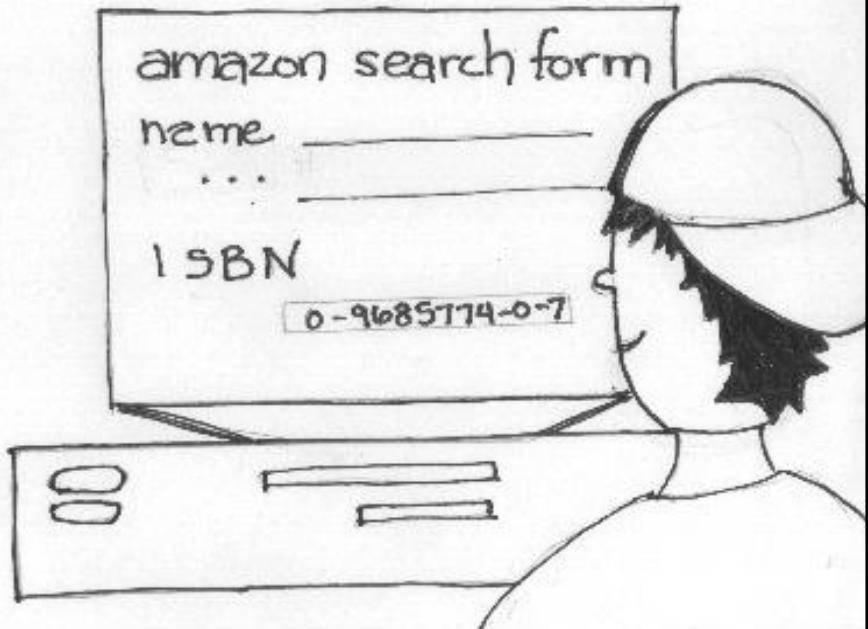
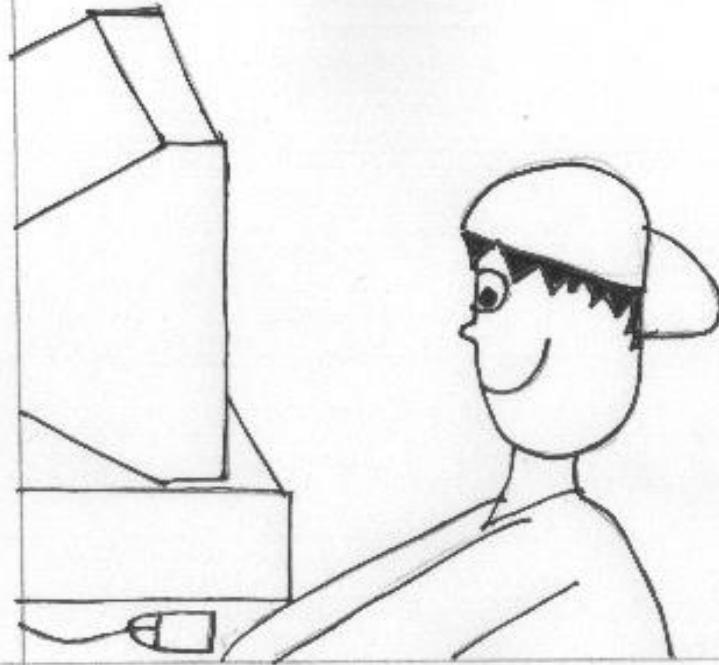
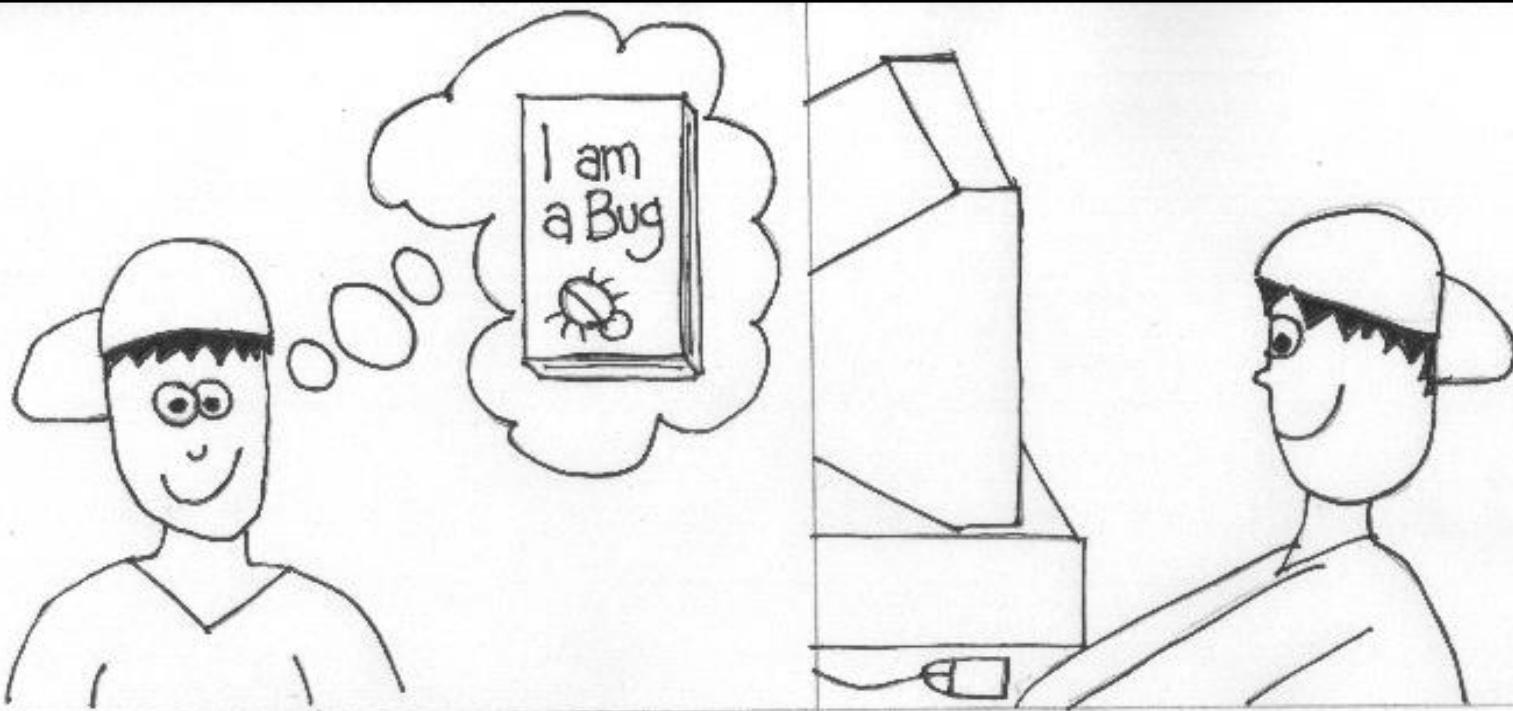
*Usage Scenarios*

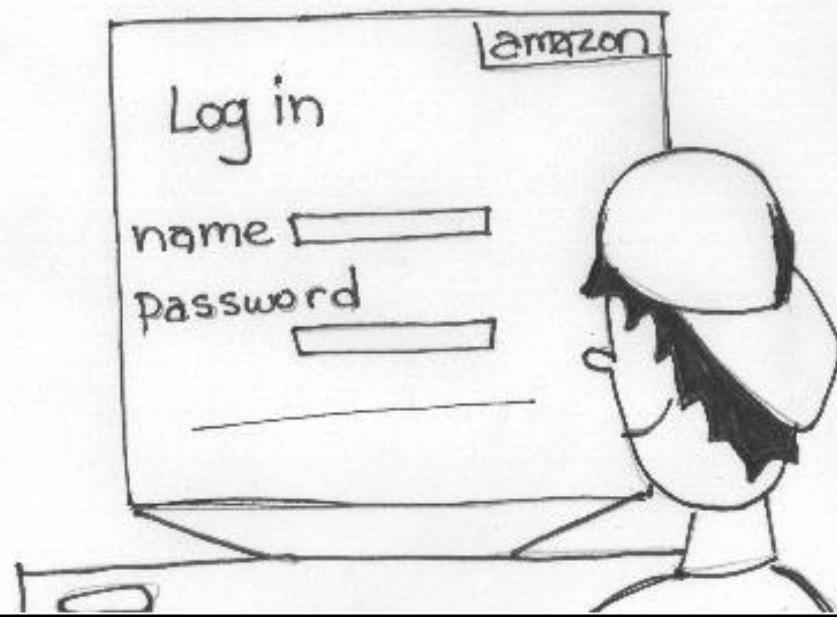
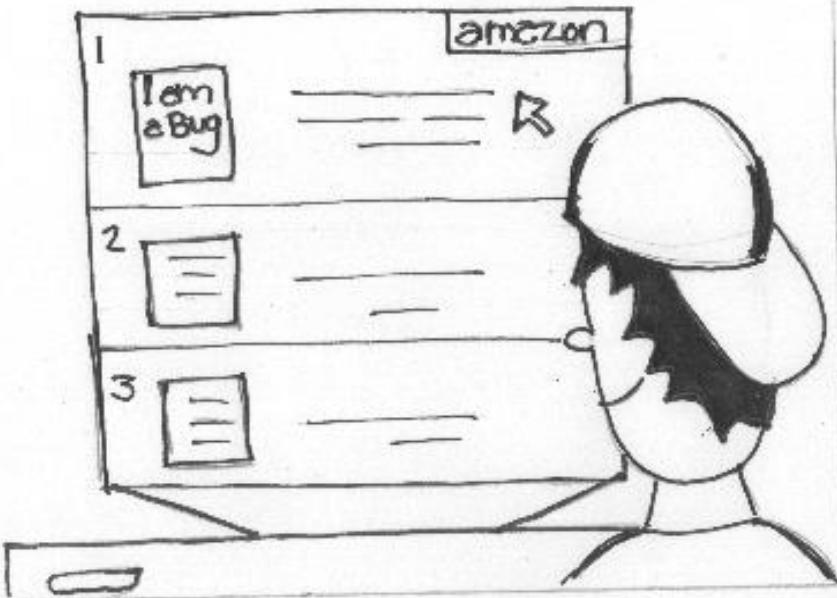
# Story Board

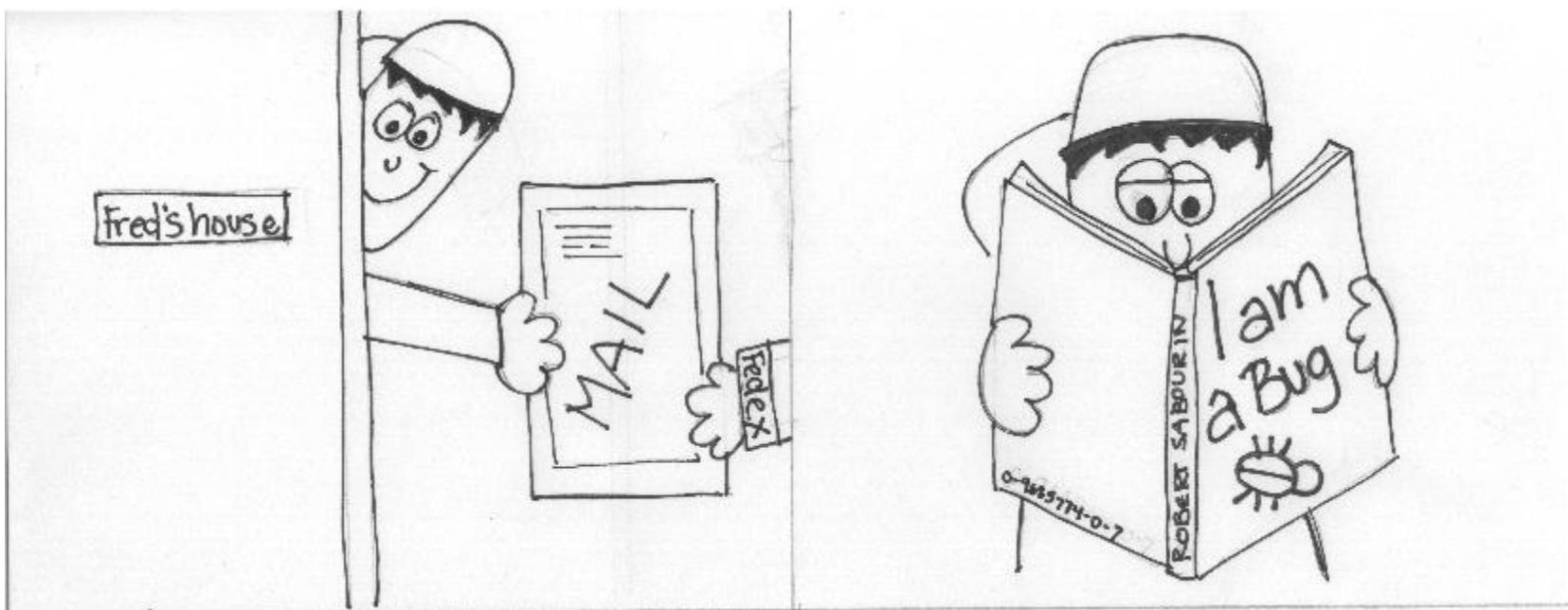
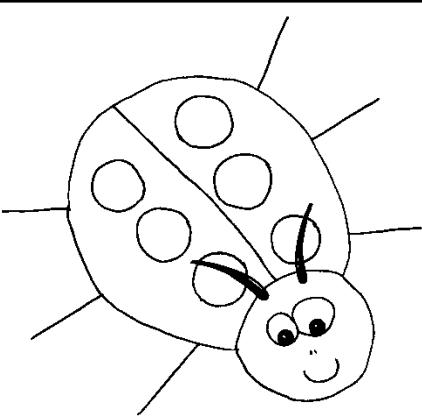


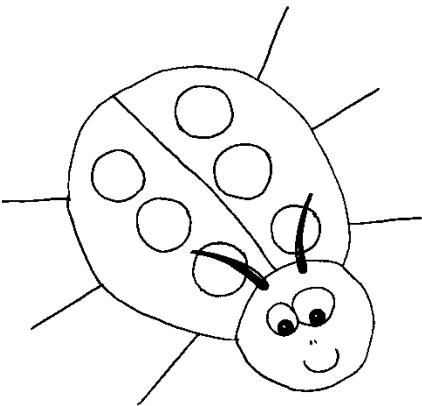
# Whiteboarding





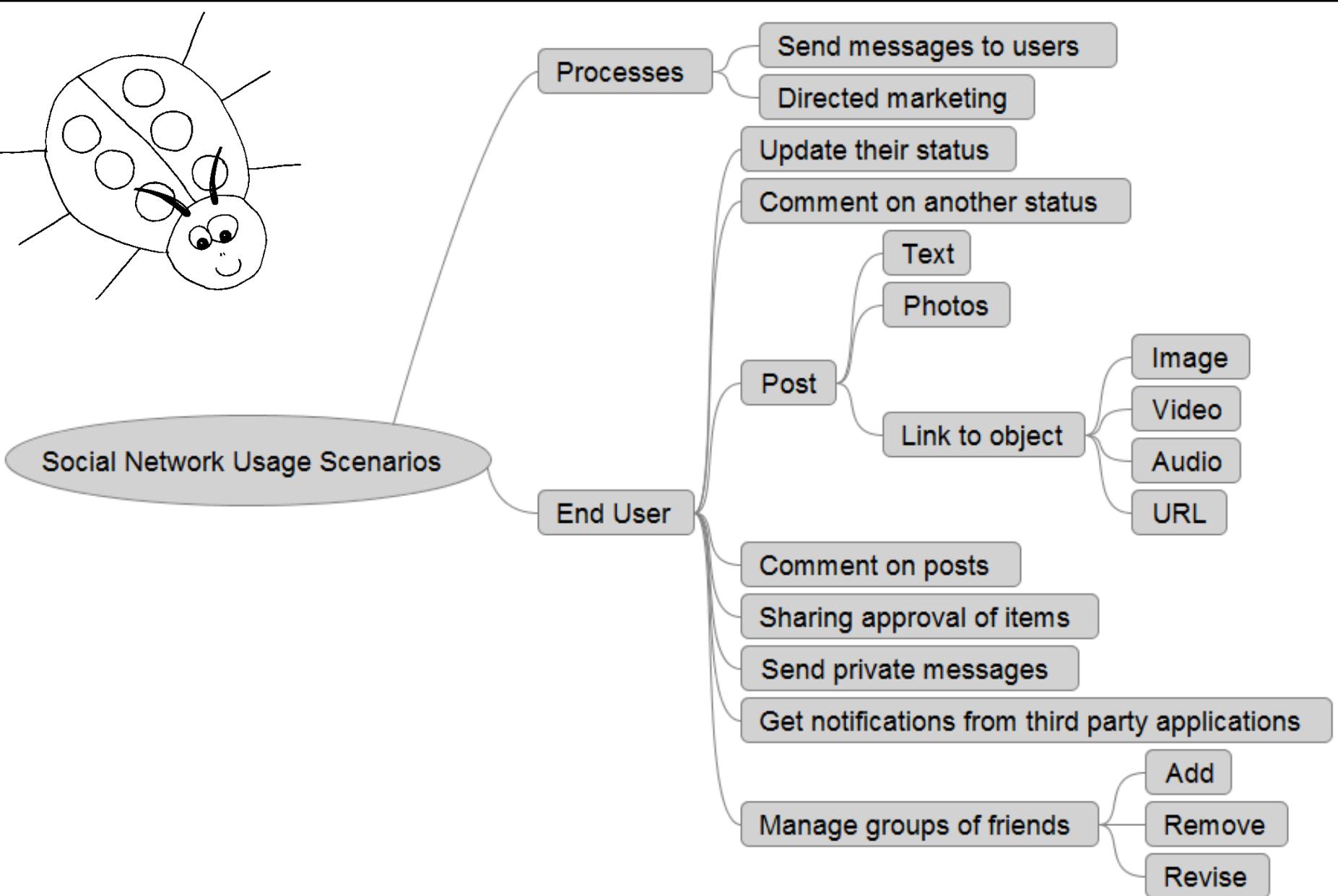


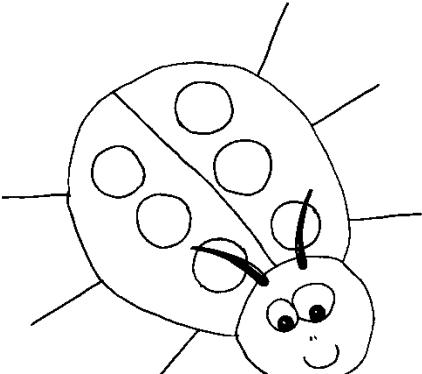




# Social Networking

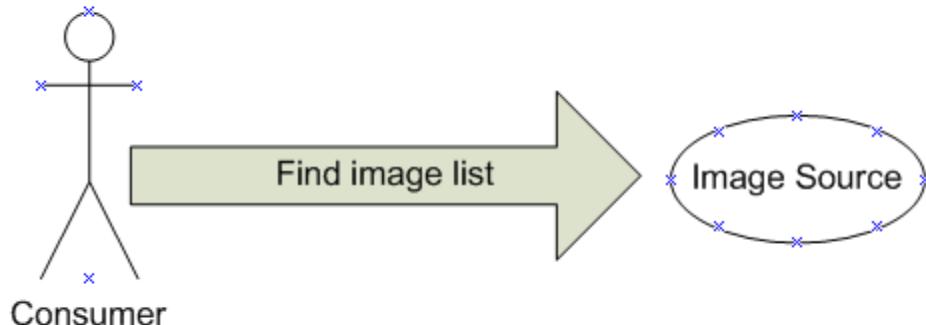
*Usage Scenarios*



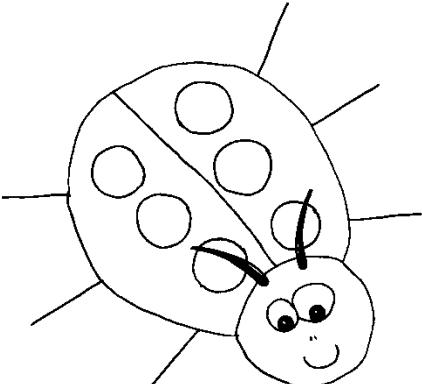


# Scenario Based Testing

## Sharing a Picture With a Group of Friends



- Image Sources
- Different sources
  - Local
  - Camera
  - Phone
  - Memory card
  - USB stick



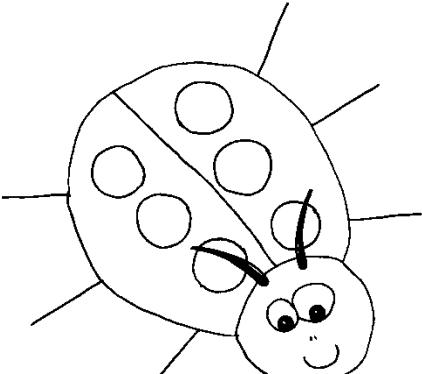
# Scenario Based Testing

## Sharing a Picture With a Group of Friends



### Images

- Image formats
- Resolutions
- Sizes
- Colors
- None
- Many

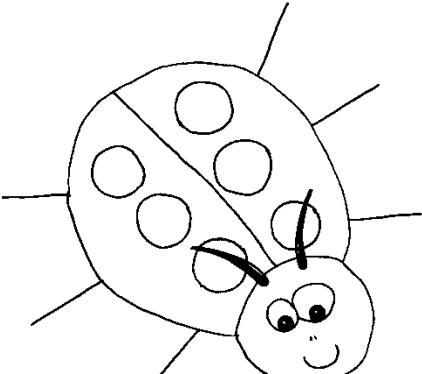


# Scenario Based Testing

## Sharing a Picture With a Group of Friends

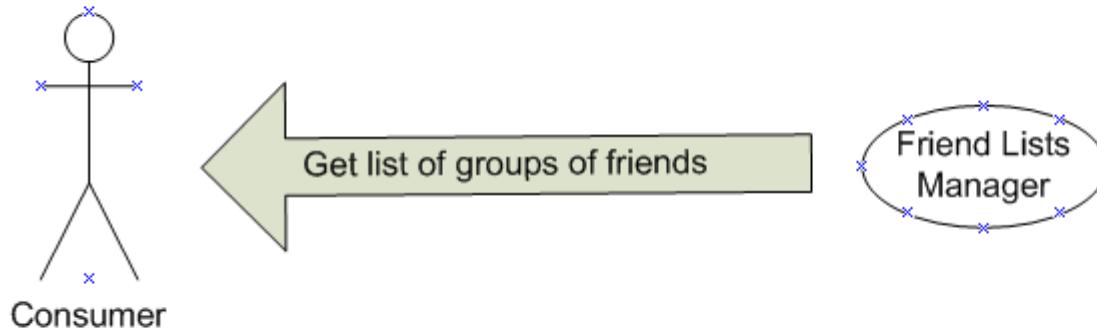


- Lists of
  - Private
  - Public

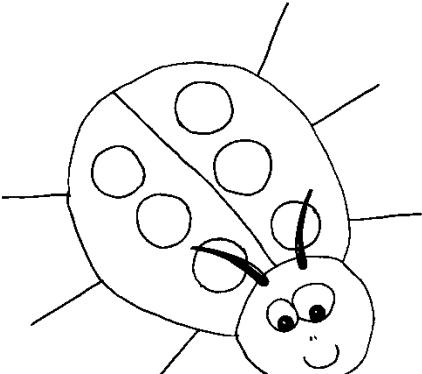


# Scenario Based Testing

## Sharing a Picture With a Group of Friends

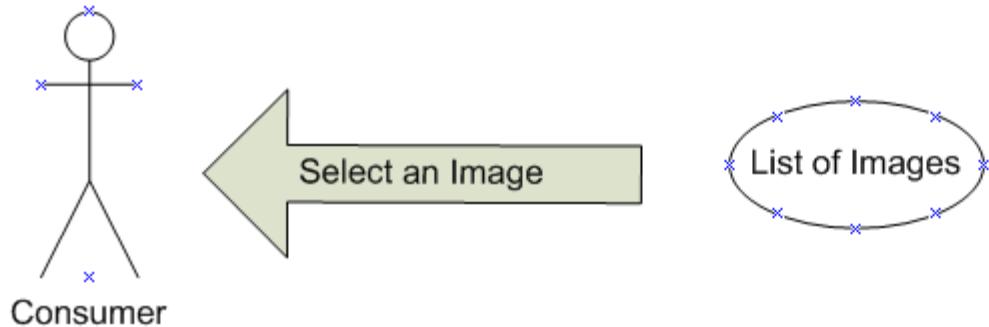


- Lists
- None
  - Many

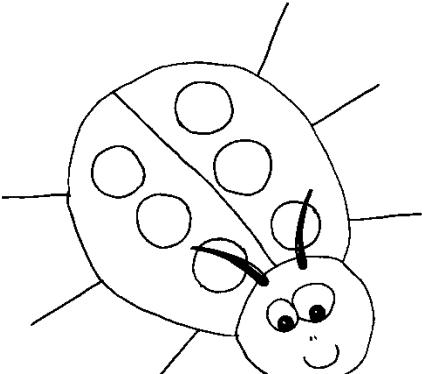


# Scenario Based Testing

## Sharing a Picture With a Group of Friends

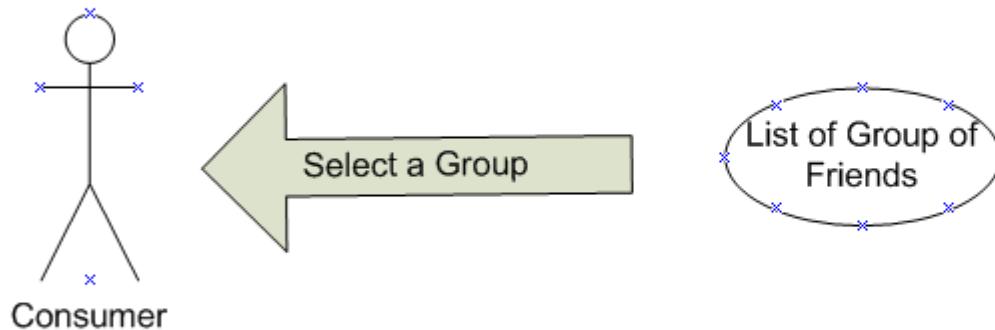


- Select by
- Name
  - Type
  - Preview
  - One
  - None
  - Many



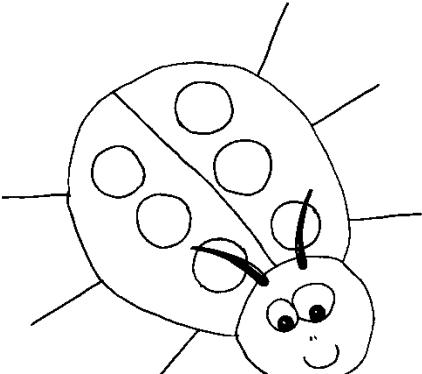
# Scenario Based Testing

## Sharing a Picture With a Group of Friends



Select by

- Name
- Membership
- Custom made for sharing this picture



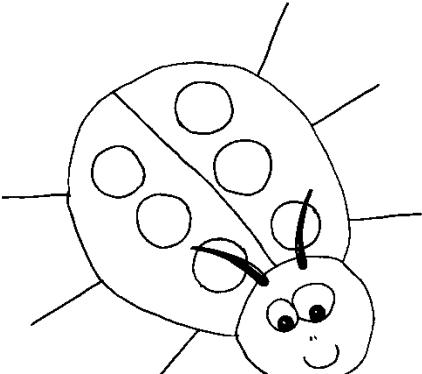
# Scenario Based Testing

## Sharing a Picture With a Group of Friends



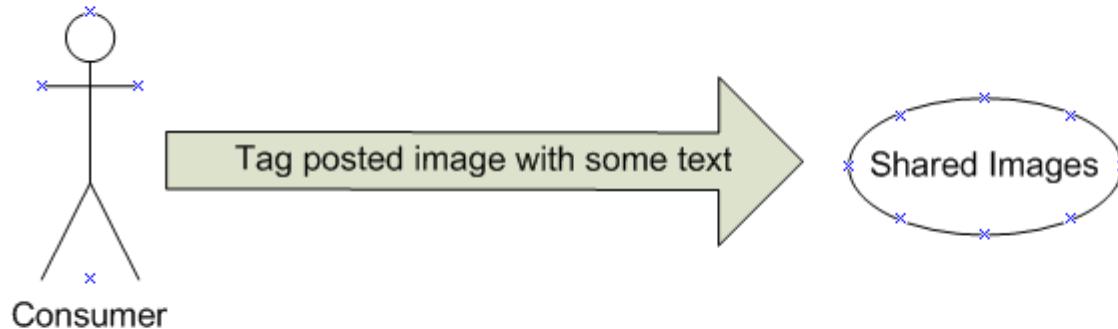
### Posted

- Single image
- Multiple images
- Single group
- Multiple groups

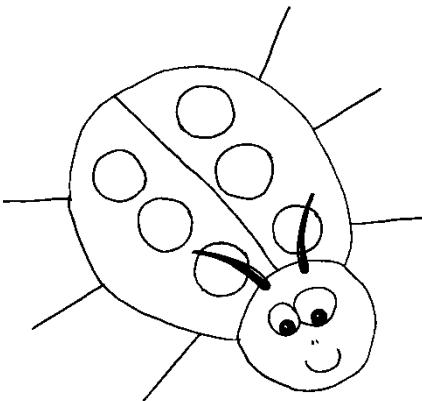


# Scenario Based Testing

## Sharing a Picture With a Group of Friends

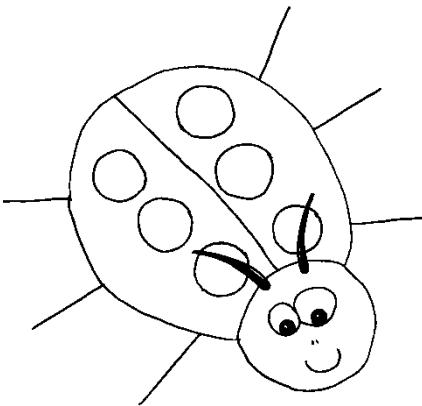


- Tagged
- No text
  - Some text
  - Link
  - Different text for different image
  - Different text for different groups



# Scenario Based Testing

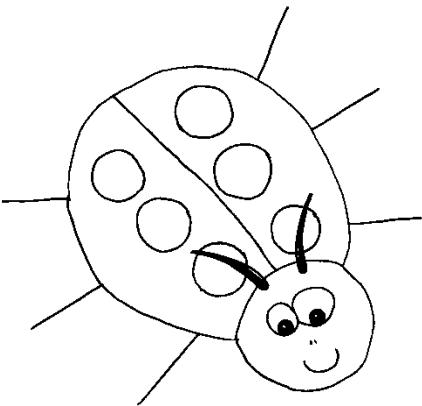
- User Experience Test Case
  - ❑ Parameterize experience
  - ❑ Walk through scenario from start to end
  - ❑ Use pre selected input for each case
  - ❑ Always run every test as if it were a user experience



# TOFT Testing

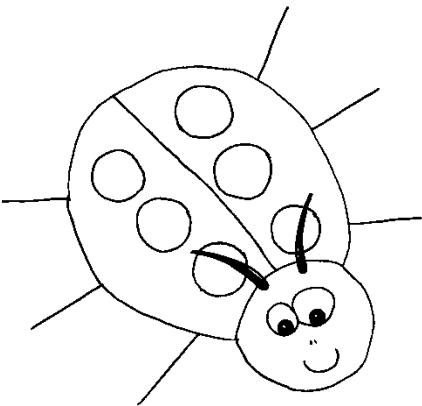
- Task Oriented Functional Testing
  - Can the user accomplish useful tasks correctly?





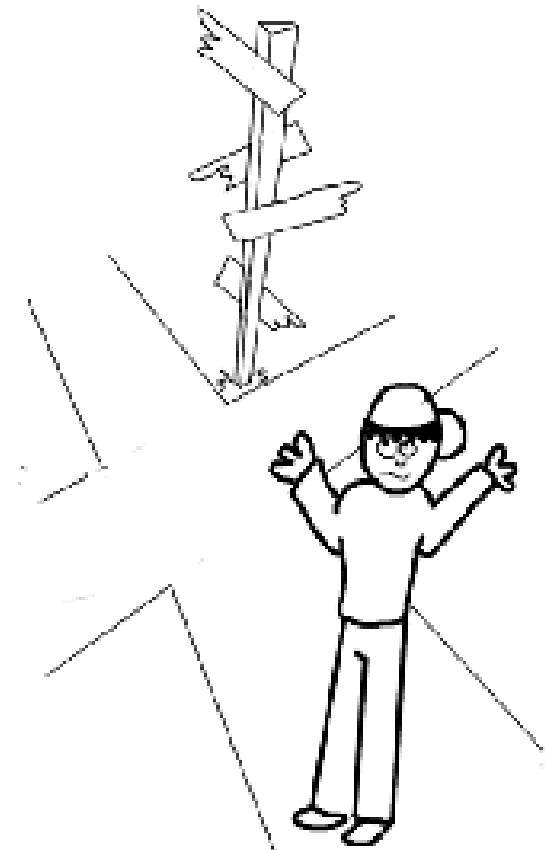
# Test Design

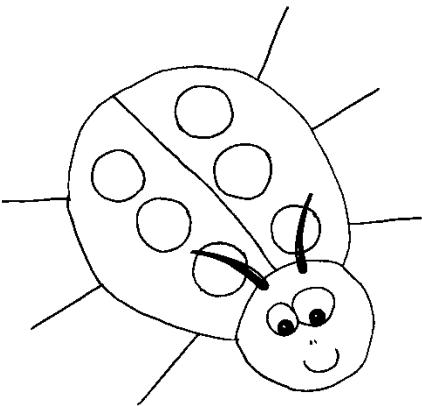
*Control Flow Testing*



# Control Flow Testing

- Exercise paths through a system
  - Data Flow
  - Transaction Flow
  - Code Flow
  - Process Flow

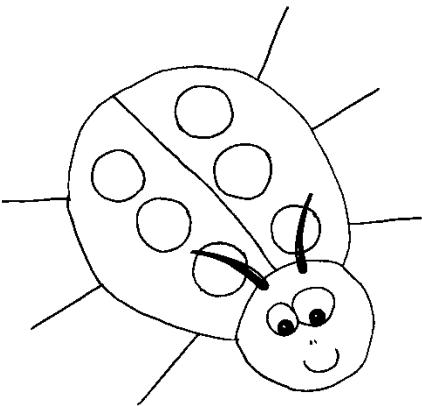




# Control Flow Testing

- Model flow
  - Create control flow diagram
  - Find basis paths
    - Minimal set of transactions
    - Exercise at least once
      - Each step
      - Each decision

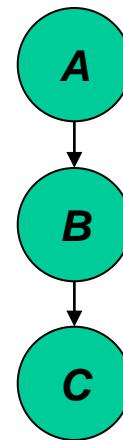


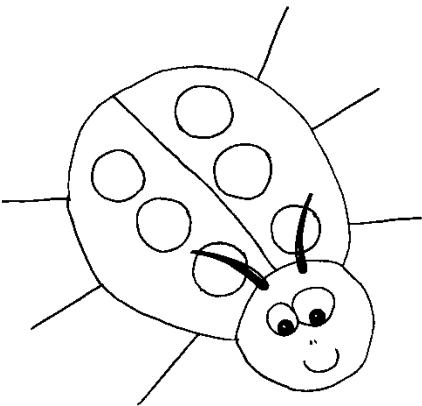


# Control Flow Testing

- Process Steps

- A flows to B
- B flow to C

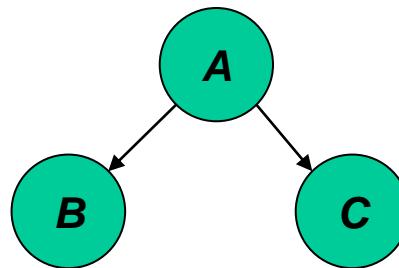


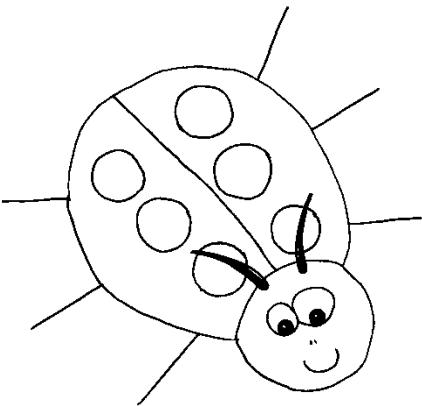


# Control Flow Testing

- Decisions

- A flow to B or C

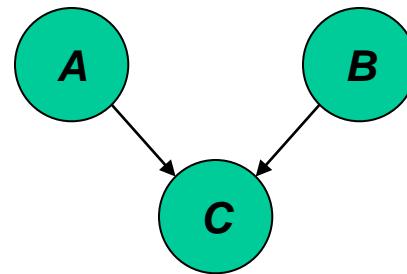


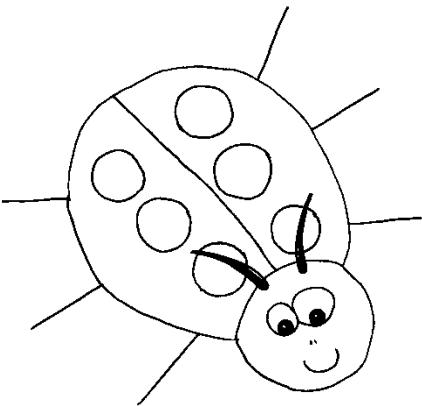


# Control Flow Testing

- Junctions

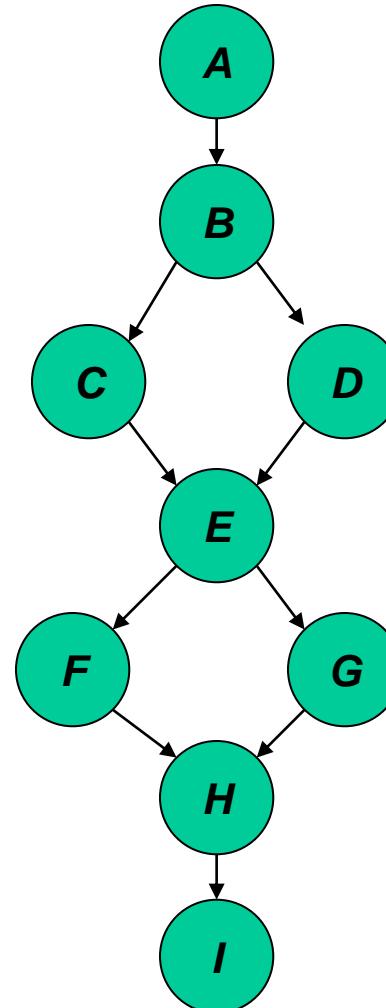
- A or B flow to C



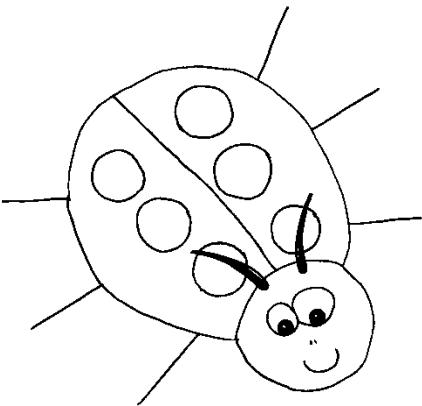


# Control Flow Testing

- Minimal basis paths
  - N – number of nodes
  - E – number of edges
  - P – number of basis paths
  - $P = E - N + 2$
  - McCabe Cyclomatic Complexity



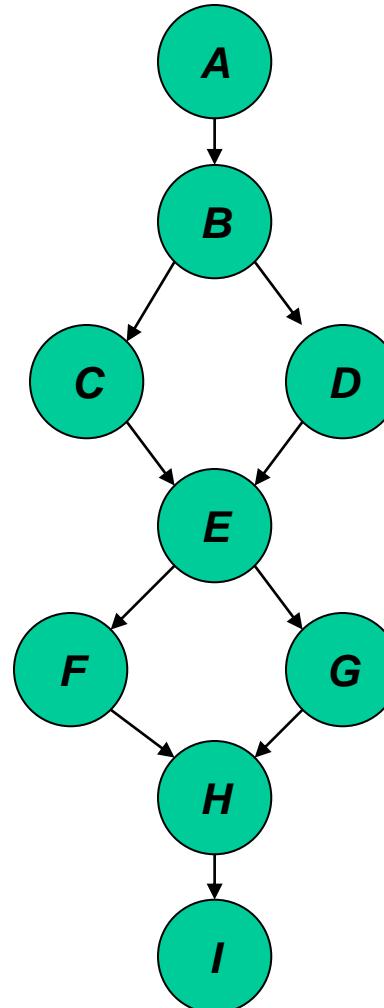
$N=9$
$E=10$
$P=10-9+2$
$P=3$



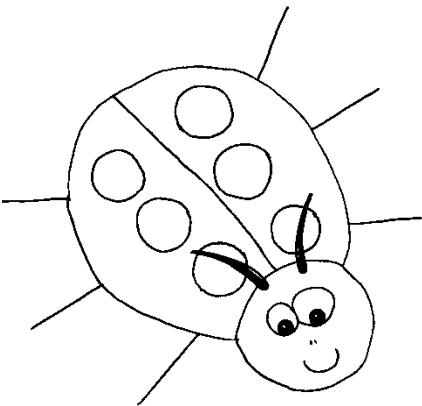
# Control Flow Testing

– A set of basis paths:

- A B D E G H I
- A B C E G H I
- A B D E F H I

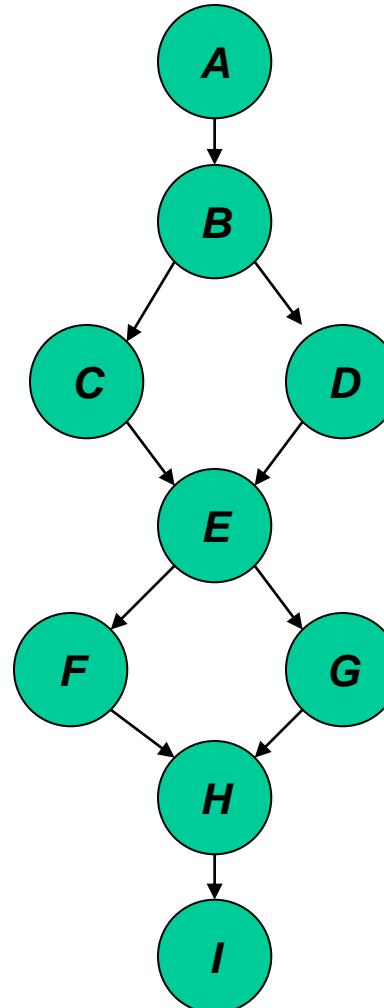


$$\begin{aligned}N &= 9 \\E &= 10 \\P &= 10 - 9 + 2 \\P &= 3\end{aligned}$$

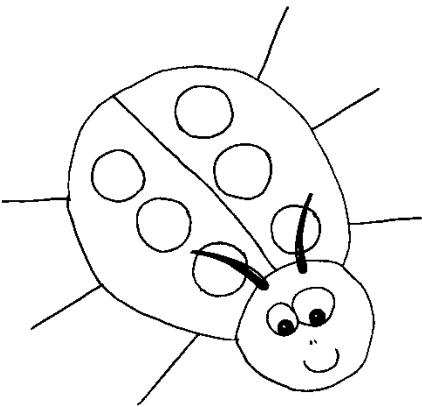


# Control Flow Testing

- Finding basis paths:
  1. Start with a *typical* baseline
  2. Flip first decision keep rest as similar as possible
  3. Continue flipping decisions on baseline
  4. After all decisions on baseline have been flipped continue on next path
  5. Stop when all paths have been exhausted



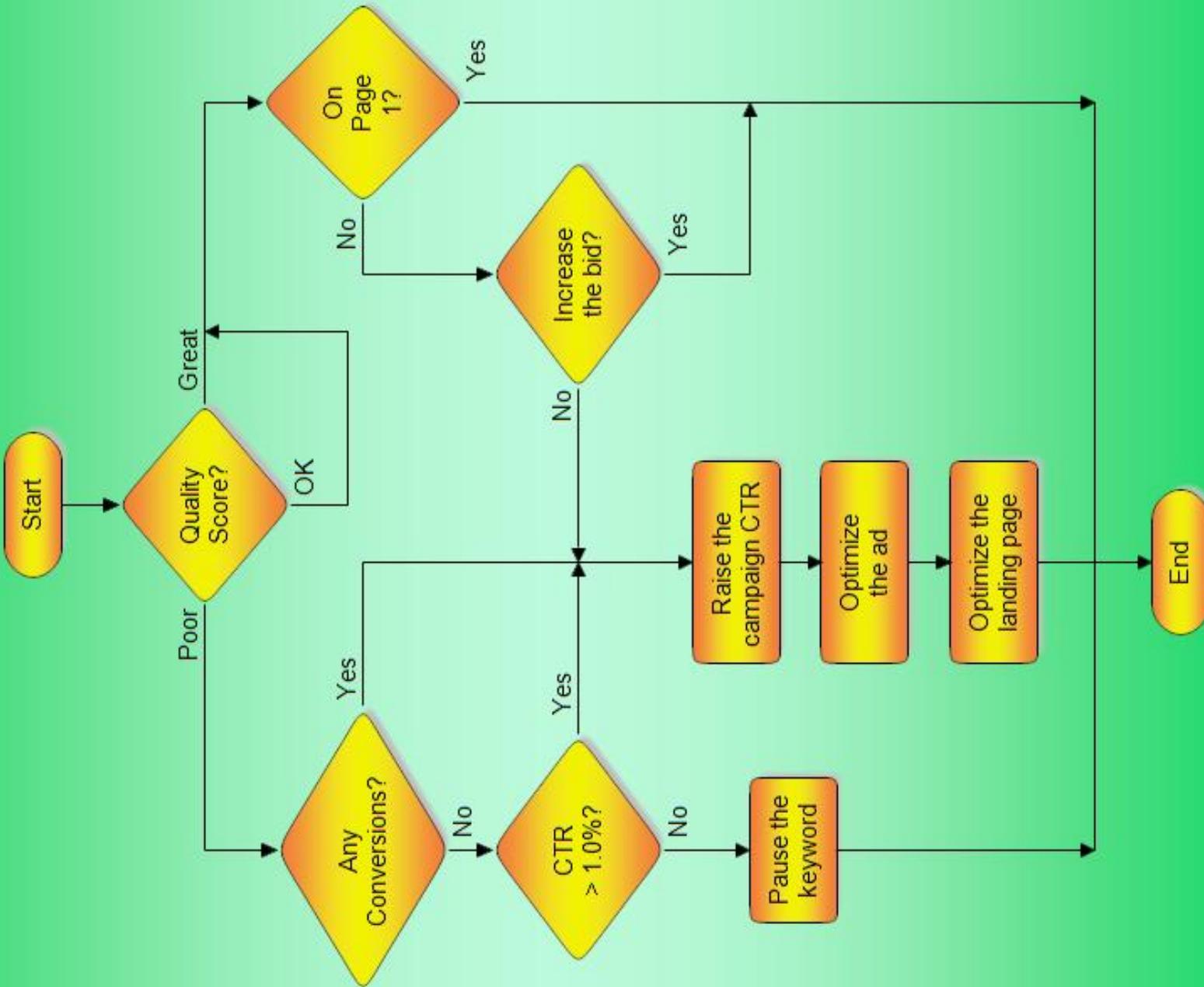
$N=9$
$E=10$
$P=10-9+2$
$P=3$



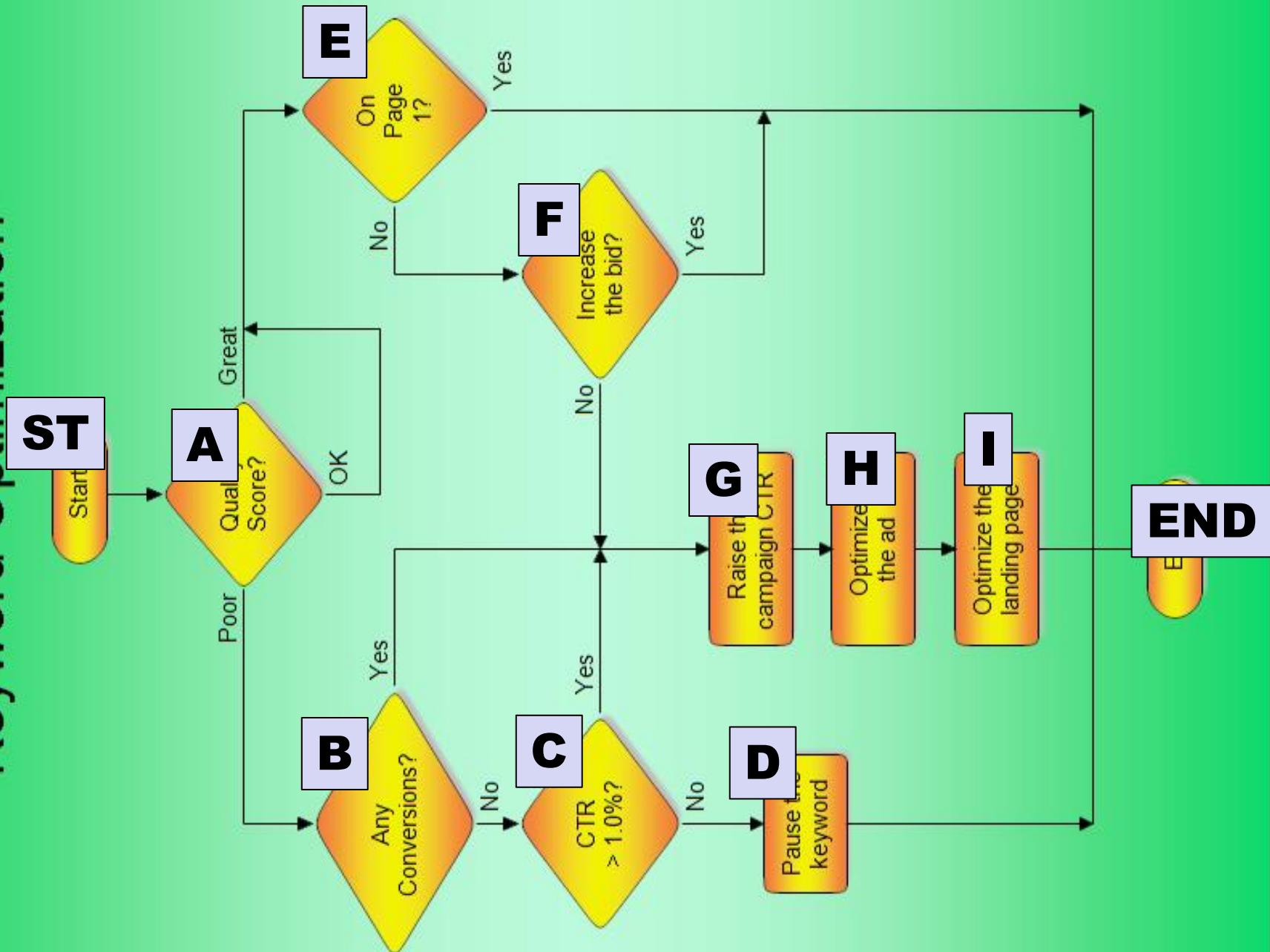
# Keyword Optimization

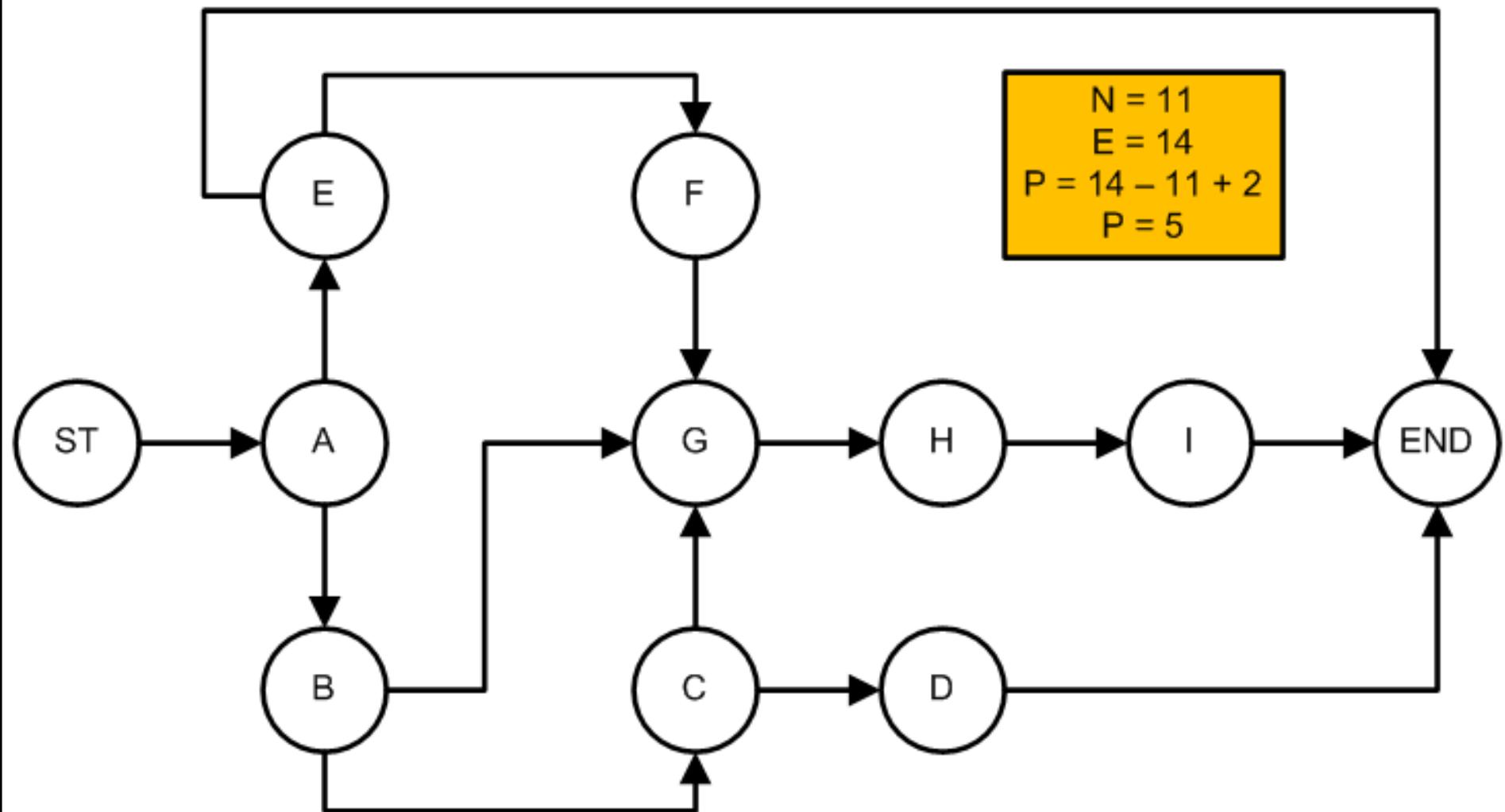
*Control Flow Diagram*

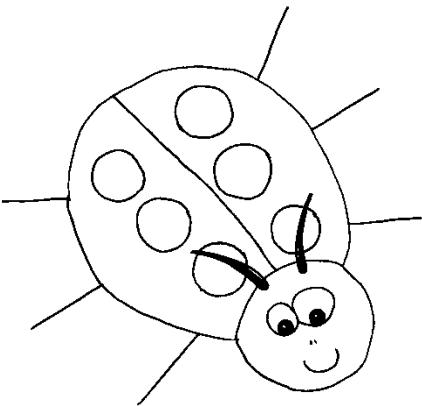
# Keyword Optimization



# Keyword Optimization



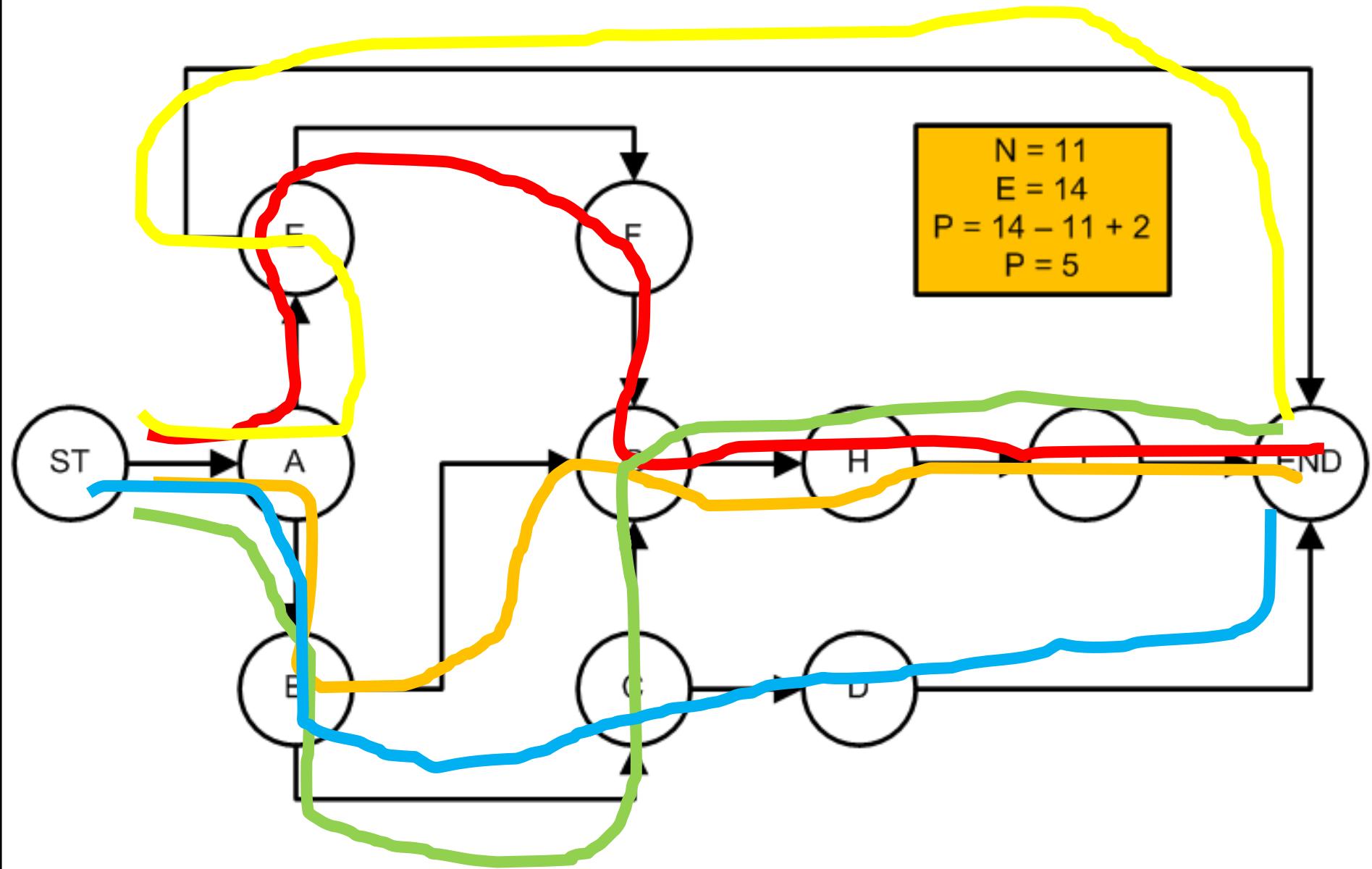


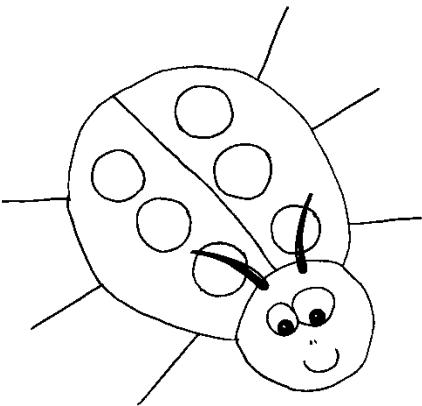


# Control Flow Testing

Basis Paths

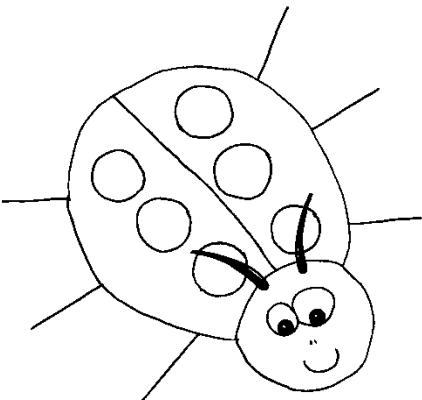
1	ST	A	E	F	G	H	I	END
2	ST	A	B	G	H	I	END	
3	ST	A	E	END				
4	ST	A	B	C	G	H	I	END
5	ST	A	B	C	D	END		





# Insurance Workflow

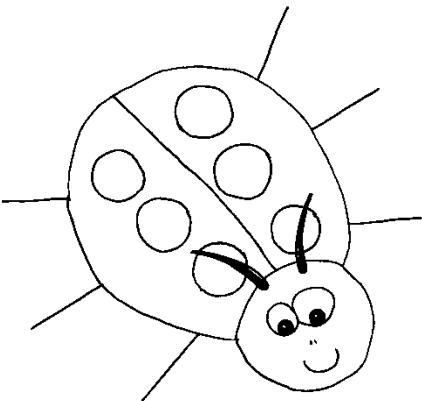
*Control Flow Diagram*



# Control Flow Testing

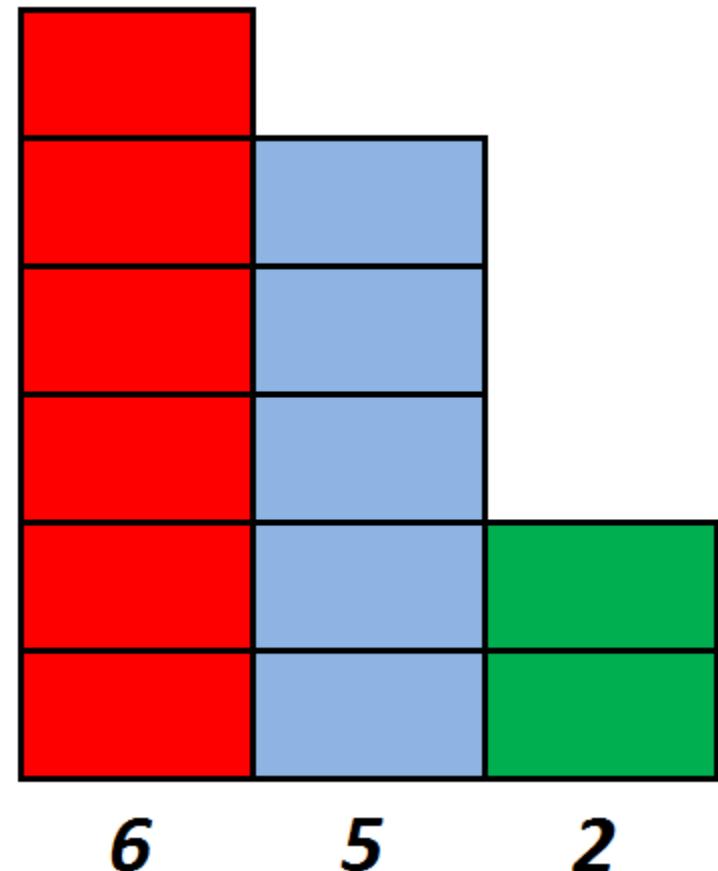
- Application screens are selected with three controls:
  - (a) has 5 options
  - (b) has 6 options
  - (c) has 2 options
- How many screens can a user choose?

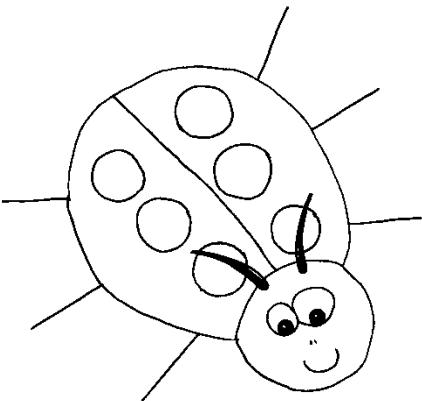




# Control Flow Testing

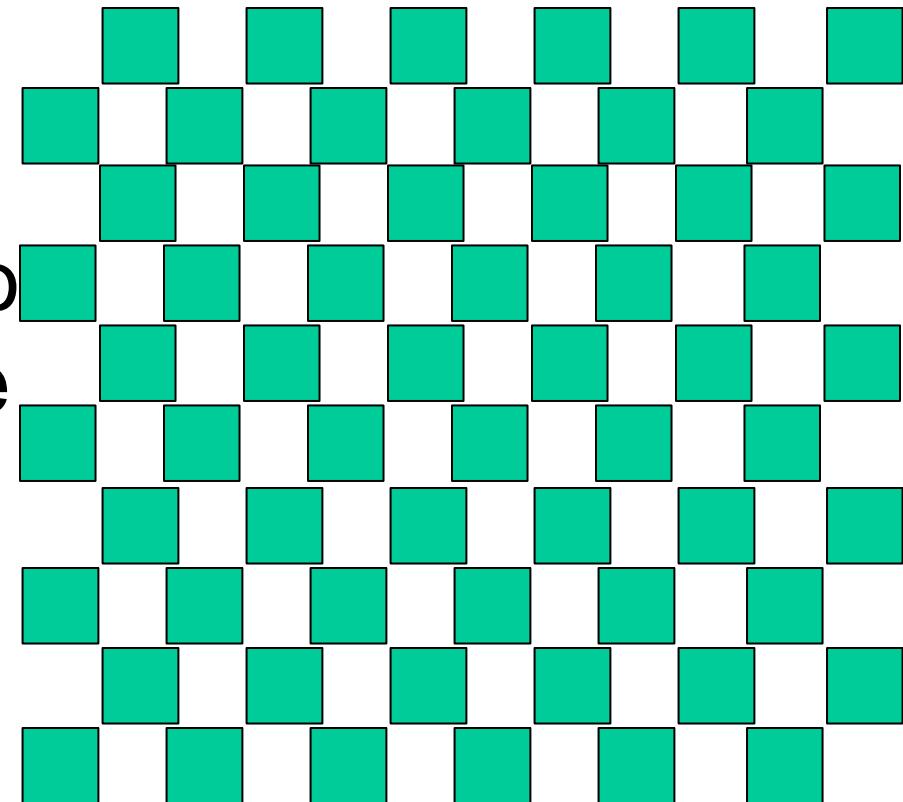
- Total Combinations  
 $= 6 \times 5 \times 2 = 60$
- To exercise each combination once a total of 60 tests would be required.

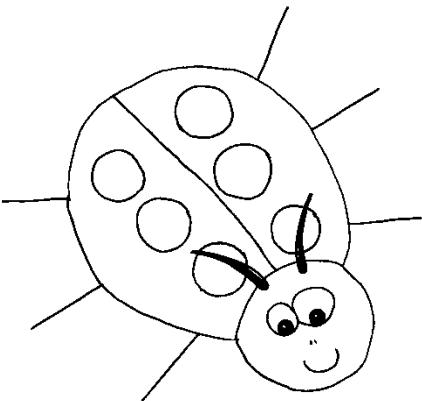




# Control Flow Testing

- How many tests would be required to exercise all possible screens in every possible order?.





# Control Flow Testing

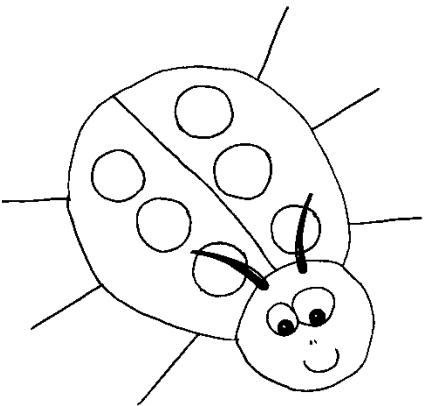
- To exercise all screens in every possible order would require

60! Test cases

$$\underline{60! = 60 \times 59 \times 58 \times \dots \times 3 \times 2 \times 1}$$

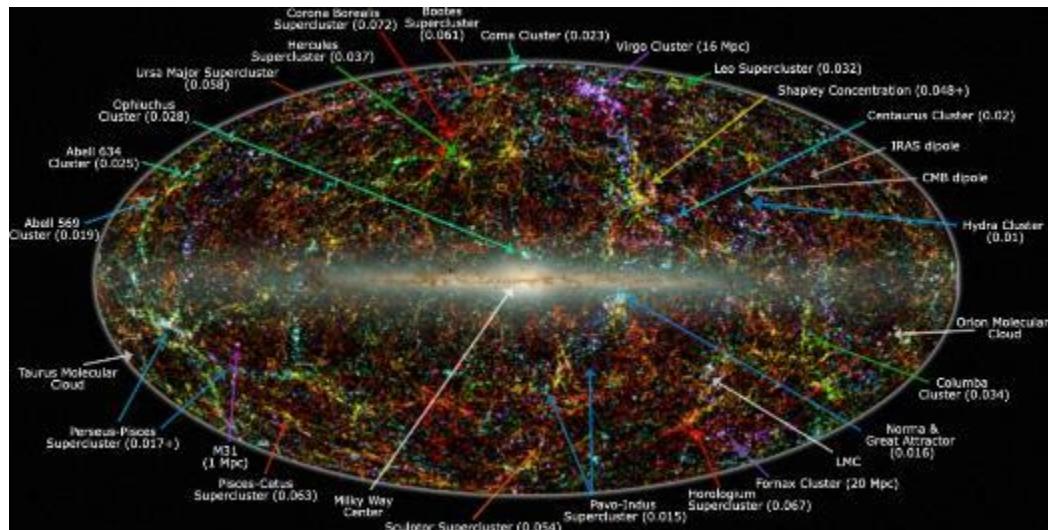
$$\underline{60! \approx 8.32 \times 10^{**81}}$$

$$n! > \sqrt{2\pi n} \left(\frac{n}{e}\right)^n.$$



# Control Flow Testing

How many atoms are in the observable universe?

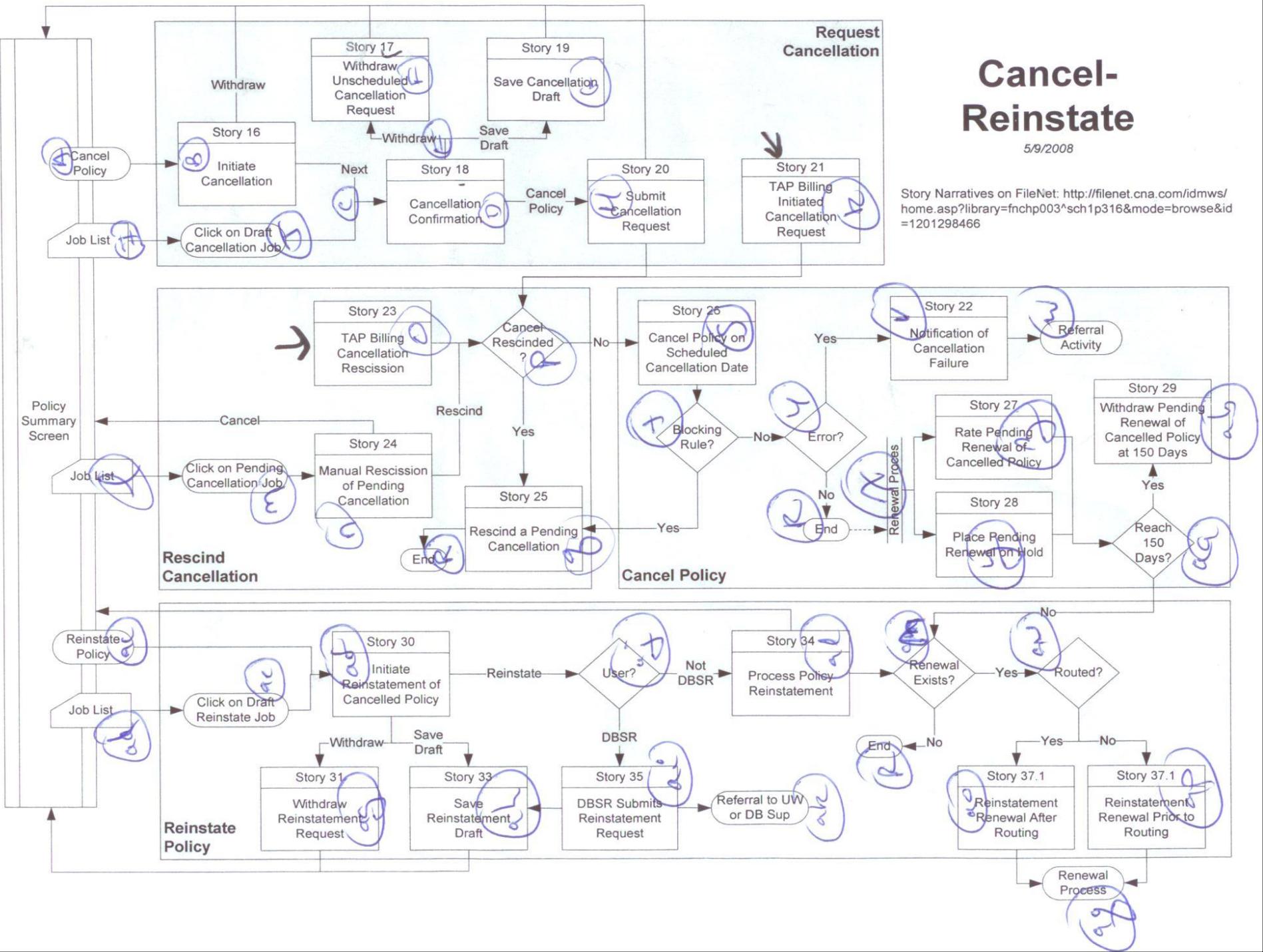


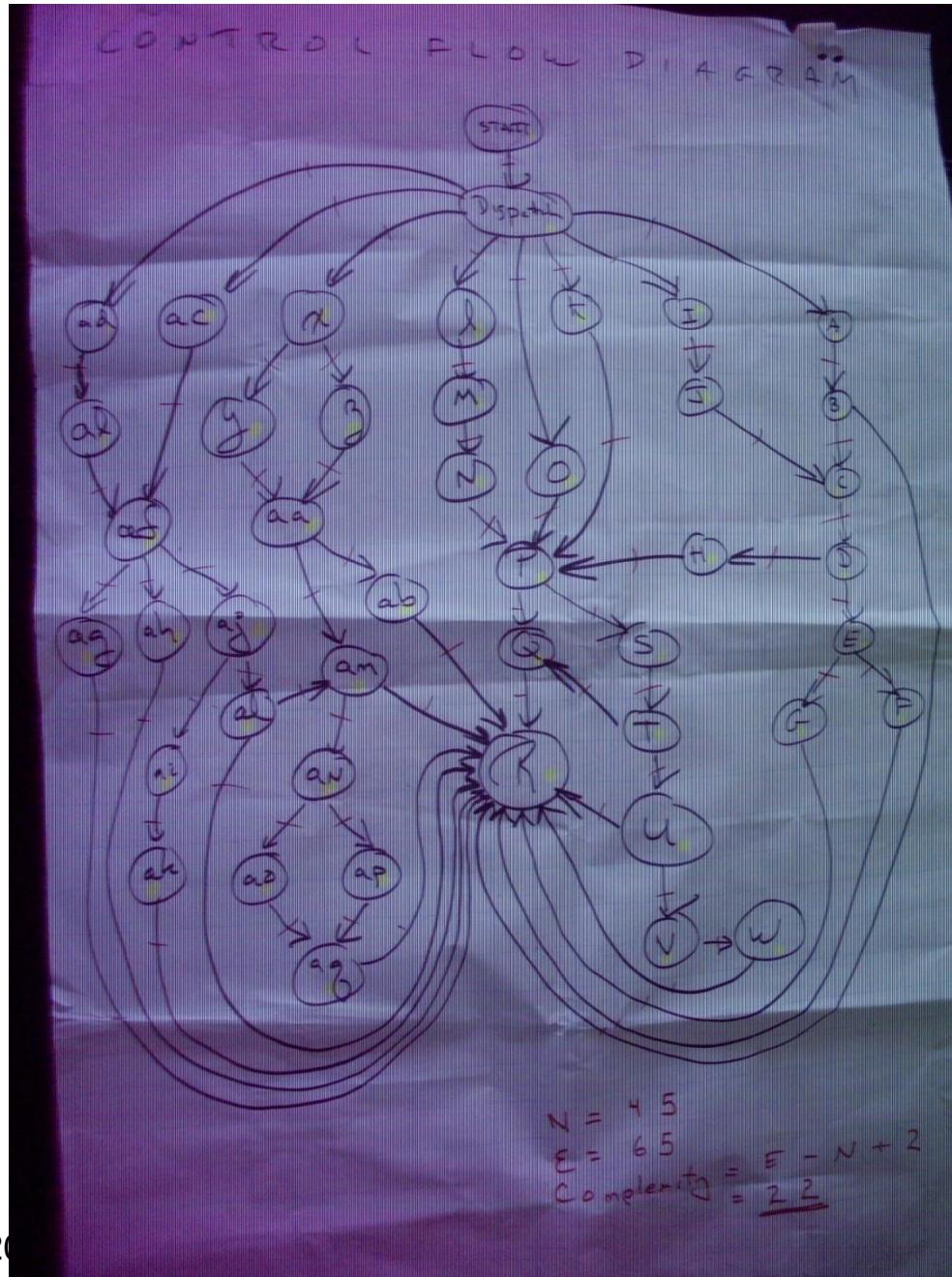
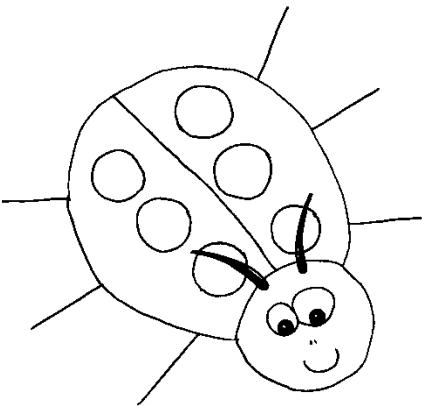
*From  $7.0 \times 10^{79}$*   
*To  $1.5 \times 10^{82}$*

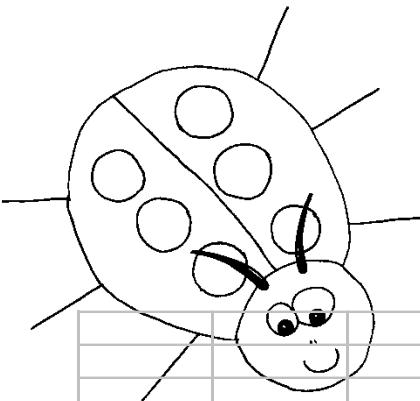
# Cancel-Reinstate

5/9/2008

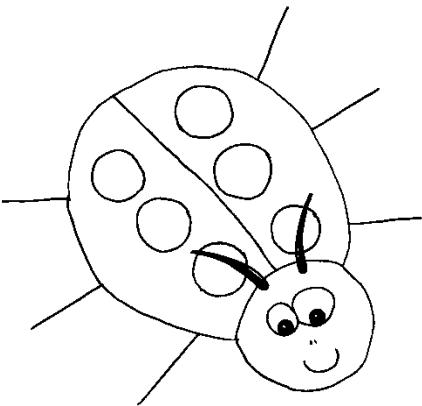
Story Narratives on FileNet: <http://filenet.cna.com/idmws/home.asp?library=fnchp003^sch1p316&mode=browse&id=1201298466>





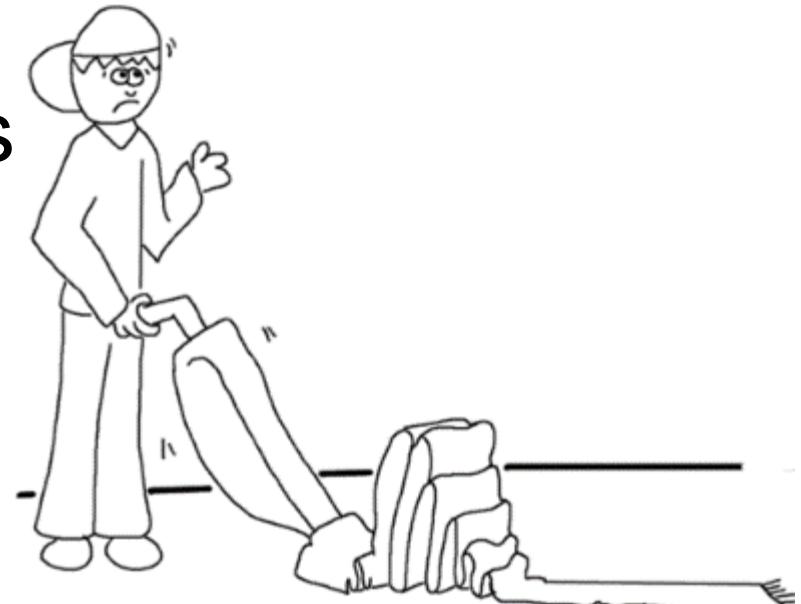


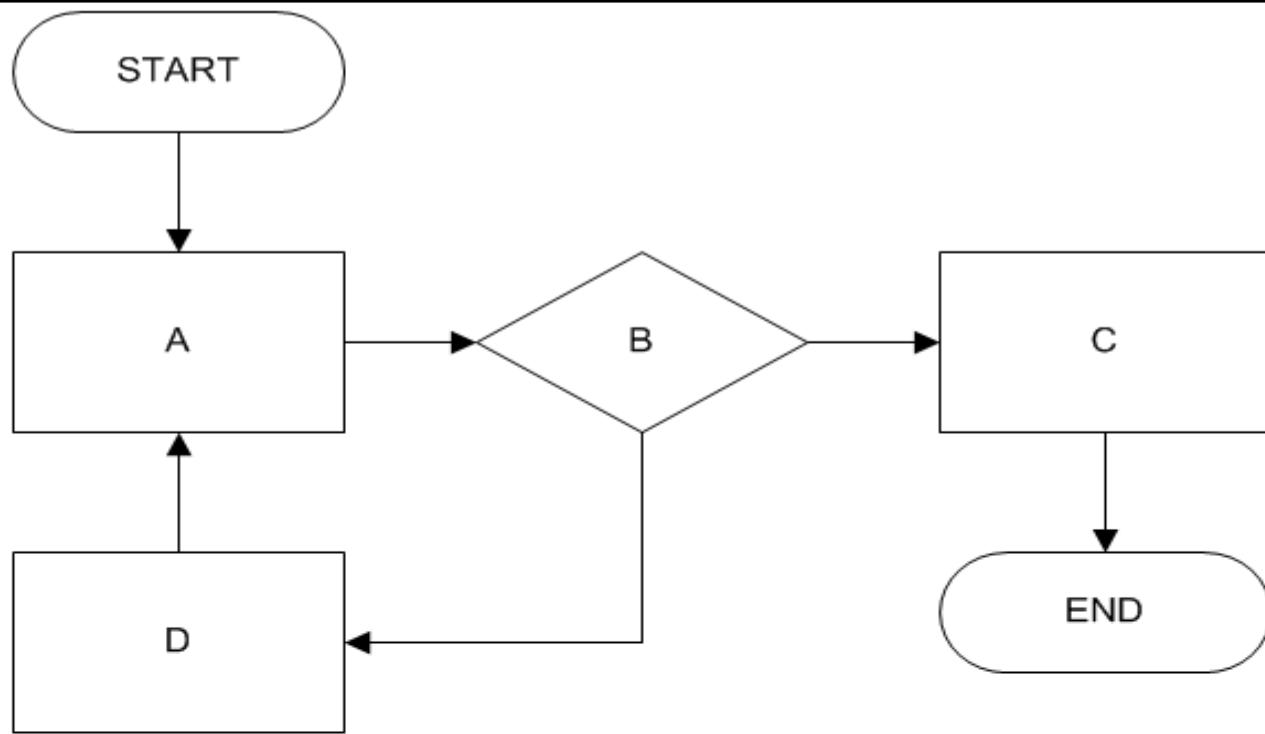
Control Flow Diagram to Identify Basis Paths									
Basis Paths									
p01	Start	Dispatch	L	M	N	P	Q	R	
p02	Start	Dispatch	X	Y	AA	AB	R		
p03	Start	Dispatch	AC	AF	AJ	AI	AK	R	
p04	Start	Dispatch	AD	AE	AF	AJ	AI	AK	R
p05	Start	Dispatch	O	P	Q	R			
p06	Start	Dispatch	K	P	Q	R			
p07	Start	Dispatch	I	J	C	D	H	P	Q
p08	Start	Dispatch	A	B	C	D	H	P	Q
p09	Start	Dispatch	L	M	N	P	S	T	Q
p10	Start	Dispatch	L	M	N	P	S	T	U
p11	Start	Dispatch	L	M	N	P	S	T	U
p12	Start	Dispatch	X	Z	AA	AB	R		
p13	Start	Dispatch	X	Z	AA	AM	R		
p14	Start	Dispatch	X	Z	AA	AM	AN	AP	AQ
p15	Start	Dispatch	X	Z	AA	AM	AN	AO	AQ
p16	Start	Dispatch	AC	AF	AG	R			
p17	Start	Dispatch	AC	AF	AH	R			
p18	Start	Dispatch	AC	AF	AJ	AL	AM	R	
p19	Start	Dispatch	AC	AF	AJ	AL	R		
p20	Start	Dispatch	I	J	C	D	E	F	R
p21	Start	Dispatch	I	J	C	D	E	G	R
p22	Start	Dispatch	A	B	R				
Nodes		45							
Edges		65							
Complexity		22 E-N+2							



# Control Flow

- Some interesting control flow test ideas
  - All nodes
  - All edges
  - All paths





Edge List:

START  
START A  
A B  
B D  
D A  
B C  
C END  
END

Basis Paths:

start node: START  
end node: END  
Number of nodes: 6 Number of edges: 6  
Cyclomatic Complexity: 2

---

START,A,B,C,END  
START,A,B,D,A,B,C,END

Number of basis paths: 2

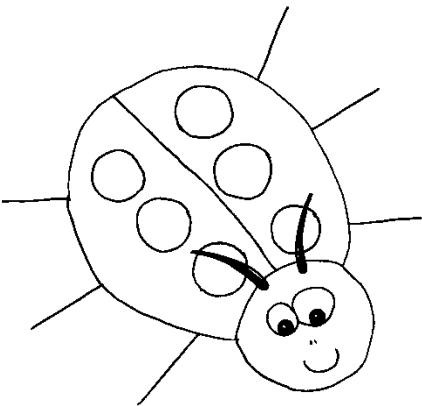
Edge Paths:

start node: START  
end node: END

---

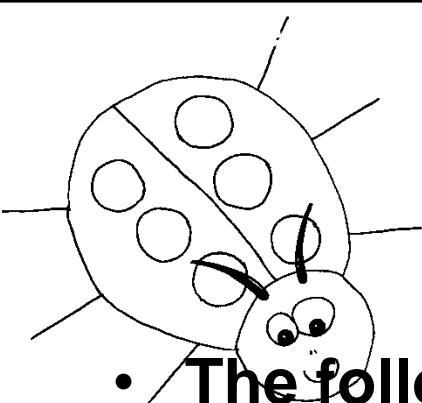
START,A,B,D,A,B,C,END

Number of paths for edge coverage: 1



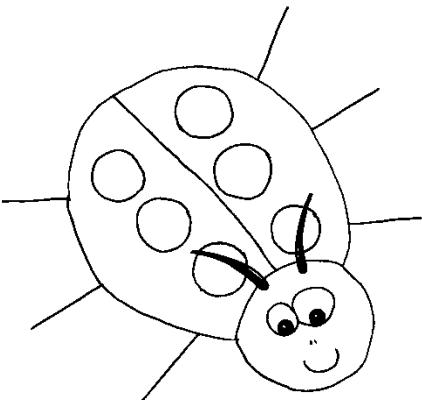
# Test Design

*Decision Tables*



- **The following instructions are taken from FAFSA<sup>SM</sup>, the Free Application for Federal Student Aid form:**

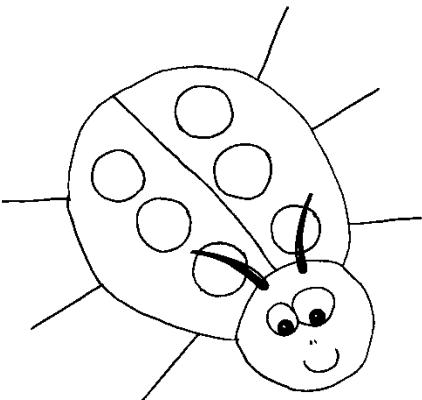
Step Four: Who is considered a parent in this step? Read these notes to determine who is considered a parent for purposes of this form. Answer all questions in Step Four about them, even if you do not live with them. Are you an orphan, or are you or were you (until age 18) a ward/dependent of the court? If Yes, skip Step Four. If your parents are both living and married to each other, answer the questions about them. If your parent is widowed or single, answer the questions about that parent. If your widowed parent is remarried as of today, answer the questions about that parent and the person whom your parent married (your stepparent). If your parents are divorced or separated, answer the questions about the parent you lived with more during the past 12 months. (If you did not live with one parent more than the other, give answers about the parent who provided more financial support during the last 12 months, or during the most recent year that you actually received support from a parent.) If this parent is remarried as of today, answer the questions on the rest of this form about that parent and the person whom your parent married (your stepparent).



# Decision Tables

## Construction

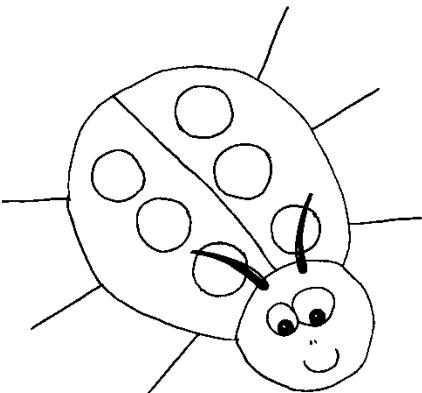
1. Identify Conditions
2. Identify Actions
3. Relate Conditions to Actions with Rules
4. Logic Reduction
5. Tests each Rule



# Decision Tables

## Logic Modeling

- Application Logic
- Business Rules
- Regulations
- Multiple Conditions
- Multiple Actions

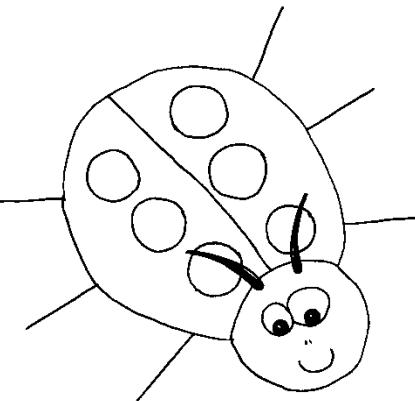


# Decision Tables

**Conditions**

**Rules**

**Actions**

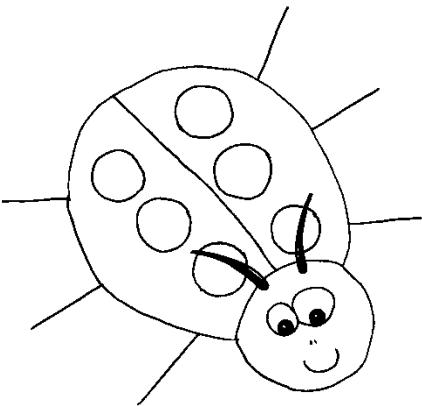


[www.wikipedia.com](https://www.wikipedia.com)

# Decision Tables

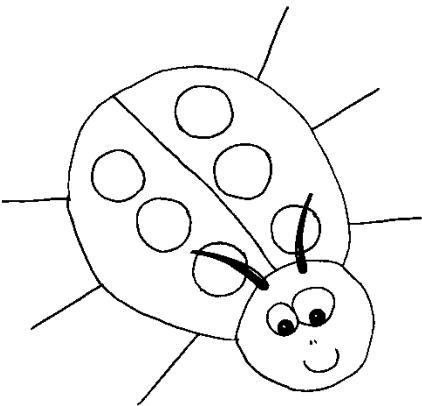
Printer troubleshooter

Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Check the power cable			X					
	Check the printer-computer cable	X		X					
	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				



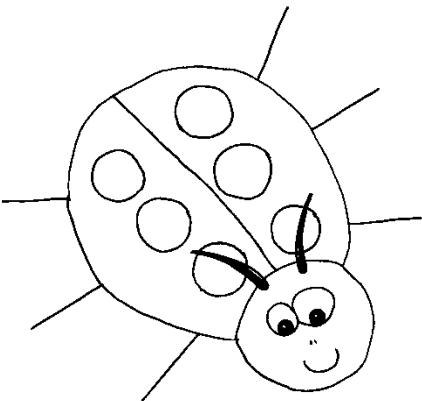
# University Acceptance

*Decision Table*



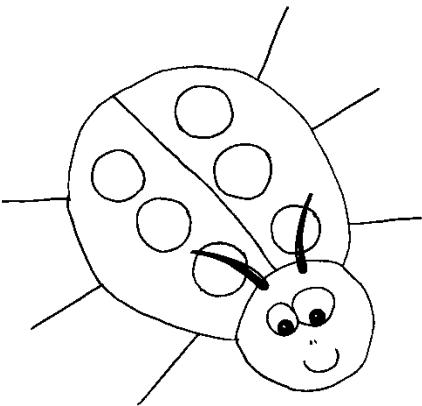
# Decision Tables Example

Conditions	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
SAT SCORE	High	High	High	High	High	High	High	High	High	High
Prerequisites	All	All	All	Core Only	Core Only	Core Only	Partial	Partial	Partial	None
Residency	In State	Out of State	Foreign	In State	Out of State	Foreign	In State	Out of State	Foreign	*
Actions										
Accept	X	X	X							
Redirect							X	X	X	X
Conditional				X	X	X				
Reject										
Grant	X	X		X	X		X	X		



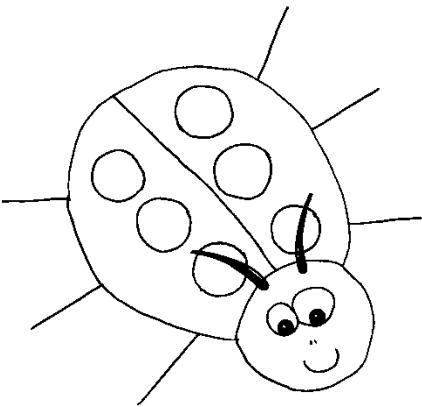
# Decision Tables Example

Conditions	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20
SAT SCORE	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium
Prerequisites	All	All	All	Core Only	Core Only	Core Only	Partial	Partial	Partial	None
Residency	In State	Out of State	Foreign	In State	Out of State	Foreign	In State	Out of State	Foreign	*
Actions										
Accept	X	X	X							
Redirect							X	X	X	
Conditional				X	X	X				
Reject										X
Grant	X			X			X			



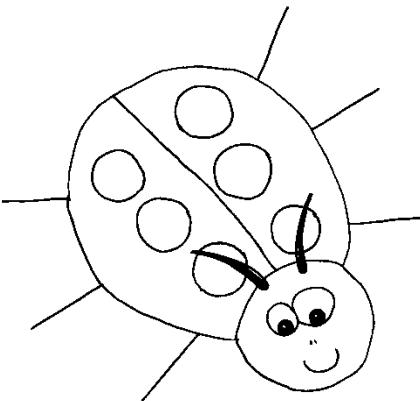
# Decision Table Example

Conditions	R21	R22	R23	R24
SAT SCORE	Low	Low	Low	Low
Prerequisites	All	Core Only	Partial	None
Residency	*	*	*	*
Actions				
Accept				
Redirect				
Conditional	X			
Reject		X	X	X
Grant				



# Customer Business Rules

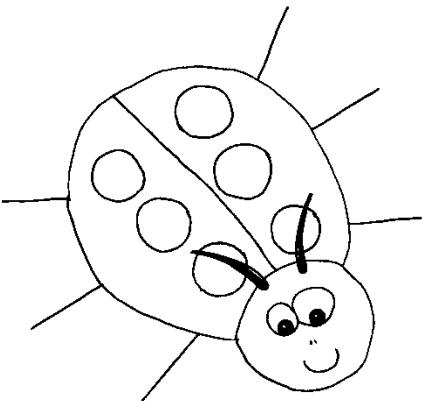
*Decision Tables*



# Decision Tables

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
<b>CONDITIONS</b>																
Roommate	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F
Account has default	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F
Pending non-default MVIN	F	T	F	T	F	T	F	T	T	T	F	F	F	T	T	F
Pending default MVIN	F	T	T	F	F	F	T	T	T	F	T	F	F	F	T	T
MVOT > MVIN	-	-	-	T	-	T	-	-	-	T	T	-	-	T	-	-
<b>ACTIONS</b>						-	-	-	-	-	-	-	-	-	-	-
Do not check "Default Move In" on MVOT	F	F	F	F	T	F	F	F	F	F	F	F	F	F	F	F
Adjust MVOT to match MVIN	F	F	F	T	F	T	F	F	F	T	T	F	F	T	F	F
Do nothing	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F
Not valid condition	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	T
Check "Default Move In" on MVOT	F	F	F	F	F	F	F	F	F	F	T	F	F	F	F	F

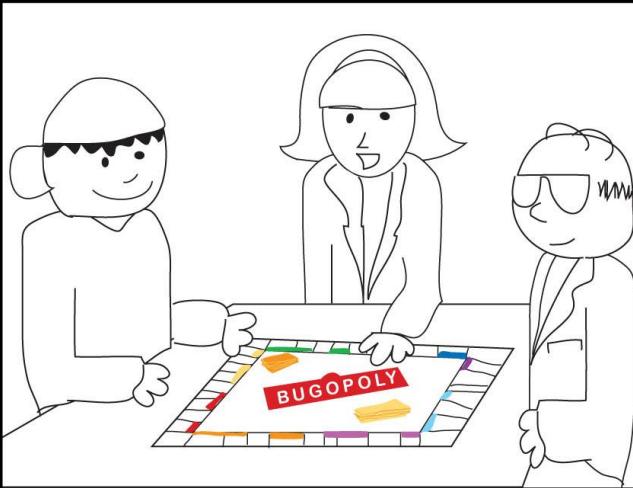
Elicited from customer



# Decision Tables

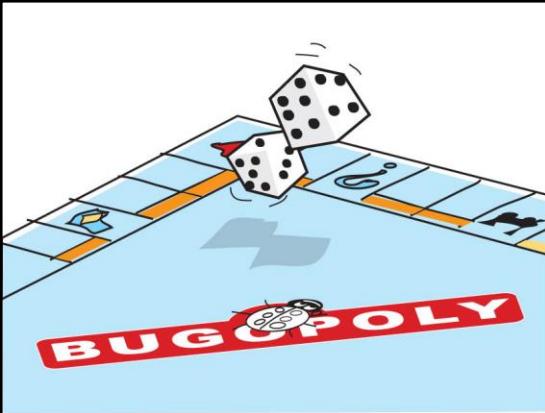
	R1	R2	R3	R4	R5	R6	R7	R8
<b>CONDITIONS</b>								
Roommate	-	-	-	-	T	T	F	F
Account has default	F	-	F	-	T	T	T	T
Pending non-default MVIN	F	T	F	T	F	F	F	F
Pending default MVIN	F	T	T	F	F	T	T	F
MVOT > MVIN	-	-	-	T	-	-	T	-
<b>ACTIONS</b>								
Do not check "Default Move In" on MVOT	F	F	F	F	T	F	F	F
Adjust MVOT to match MVIN	F	F	F	T	F	F	T	F
Do nothing	T	F	F	F	F	T	F	F
Not valid condition	F	T	T	F	F	F	F	F
Check "Default Move In" on MVOT	F	F	F	F	F	F	F	T

Reduced



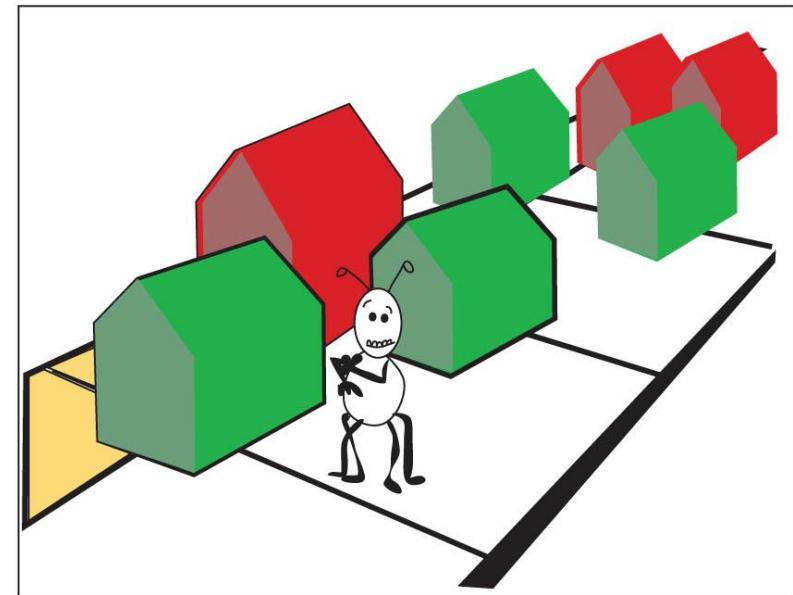
# Monopoly® House Purchase Rules

*Decision Tables*



# Decision Tables

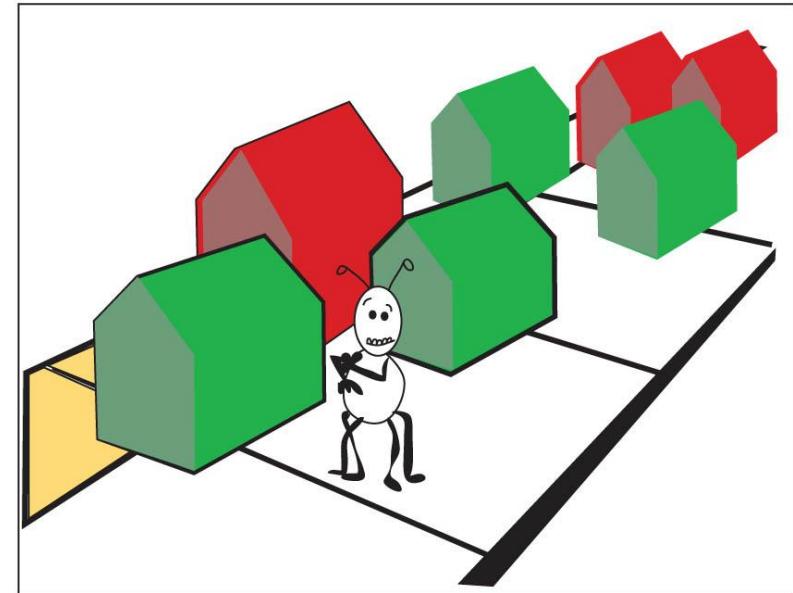
- House Purchase Rules
- Property must be in a color group
- A player must own all properties of a color group
- No properties can be mortgaged

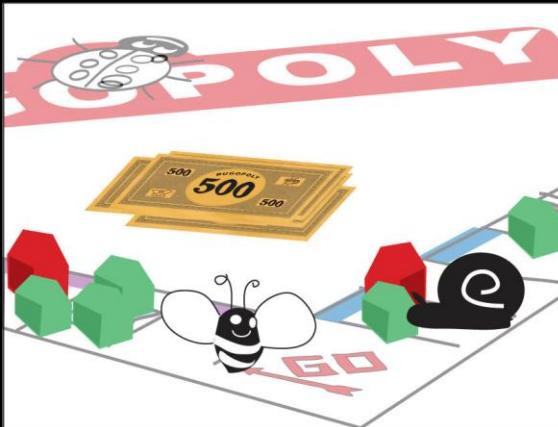




# Decision Tables

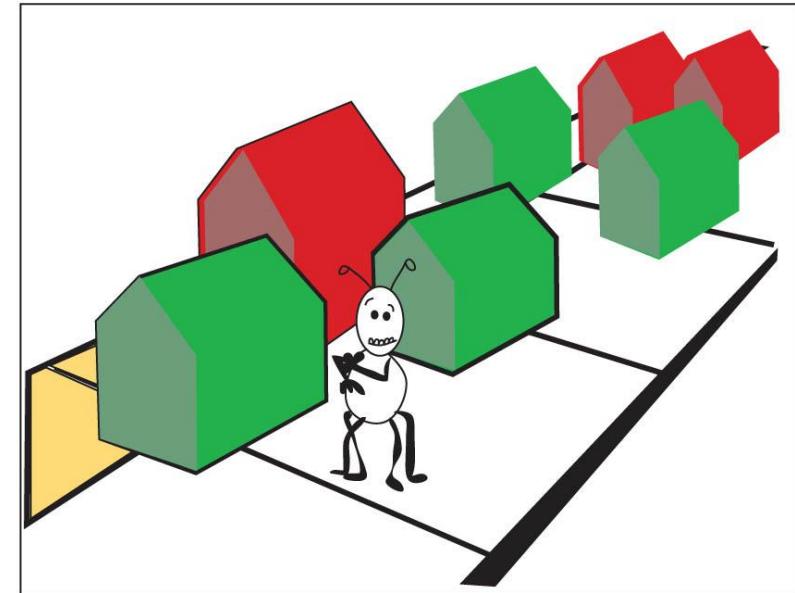
- House Purchase Rules
- Houses must be available for sale
- Houses must be built progressively
- Each property can have a maximum of 4 houses





# Decision Tables

- House Purchase Rules
- If multiple players attempt to purchase the same house the highest auction bidder gets it



As a player, I purchase a house on one of my properties, so as to increase potential rent revenues respecting the house purchase rules.

## House Purchase Rules

Conditions	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11
Property in a color group	Y	N	X	X	X	X	X	X	Y	Y	X
All color group owned by player	Y	X	N	X	X	X	X	X	Y	Y	X
Any properties of color mortgaged	N	X	X	Y	X	X	X	X	N	N	X
Houses are available for sale	Y	X	X	X	N	X	X	X	Y	Y	X
Number of houses on target property	<4	X	X	X	X	4	X	X	<4	<4	X
Other property in color group has fewer houses	N	X	X	X	X	X	Y	X	N	N	X
Hotel on target property	N	X	X	X	X	X	X	Y	N	N	X
Multiple players attempt to purchase same house	N	X	X	X	X	X	X	X	Y	Y	X
Highest bidder	X	X	X	X	X	X	X	X	Y	N	X
Sufficient funds are available for purchase	Y	X	X	X	X	X	X	X	Y	X	N
Actions											
Successful purchase	Y	N	N	N	N	N	N	N	Y	N	N
Target price	Y	N	N	N	N	N	N	N	N	N	N
Auction price	N	N	N	N	N	N	N	N	Y	N	N



# Wrap-O-Matic

*Decision Tables*



# Decision Tables

- Wrapping Rules:
  - Disallows ribbons applied to unwrapped chocolates.
  - Disallows hollow chocolates tied with metallic ribbon.
  - Uses the gentle wrapping algorithm with tissues wrappers.



# Decision Tables

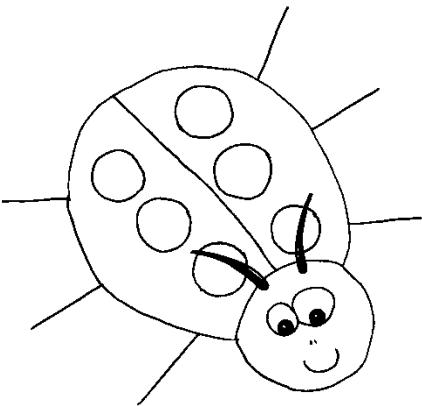
- Wrapping Rules:
  - Uses the rapid wrapping algorithm whenever chocolates do not have ribbons and do not have tissue wrappers.
  - Uses the gentle algorithm whenever hollow chocolates are tied with ribbons.
  - Uses the normal algorithm for all other cases.



# Decision Tables

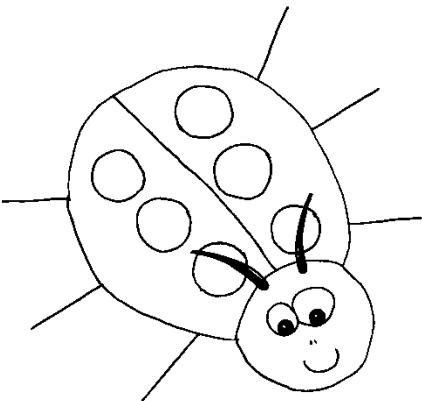
		Rules																
		R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	R14	R15		
Conditions	Viscosity	Hollow										Not Hollow						
	Ribbon	Metallic			Other			None			Metallic or Other				None			
	Wrapper	Metallic or Paper	Tissue	None	Metallic or Paper	Tissue	None	Metallic or Paper	Tissue	None	Metallic or Paper	Tissue	None	Metallic or Paper	Tissue	None		
Actions	Disallow	x	x	x	.	.	x	.	.	.	.	.	x	.	.	.	.	
	Rapid Algorithm	.	.	.	-	.	-	x	.	x	.	.	.	x	.	.	x	
	Normal Algorithm	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	
	Gentle Algorithm	.	-	.	x	x	x	.	x	.	.	x	.	.	.	x	.	

## Wrapping Rules



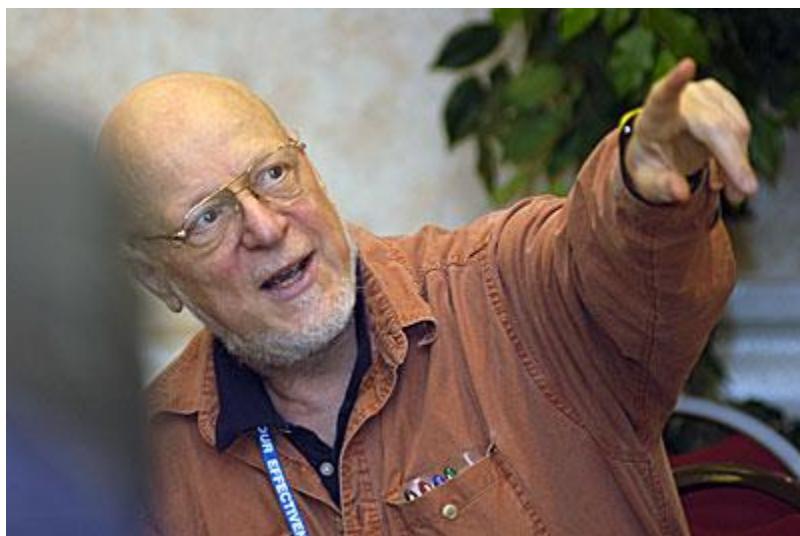
# Test Design

*State Models*

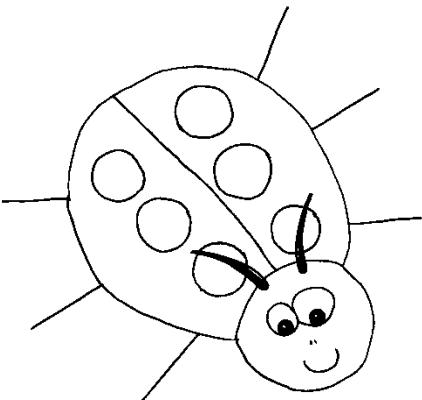


# Gerald M. Weinberg

“A state is a situation which can be  
recognized if it occurs again”



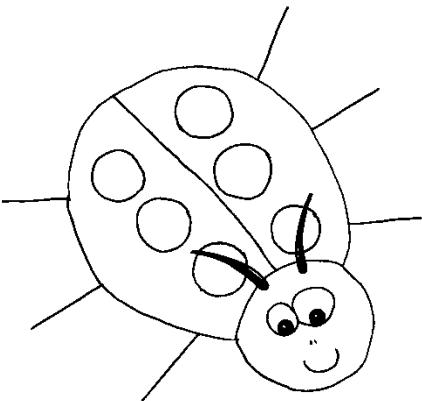
An Introduction to  
General Systems  
Thinking  
Dorset House



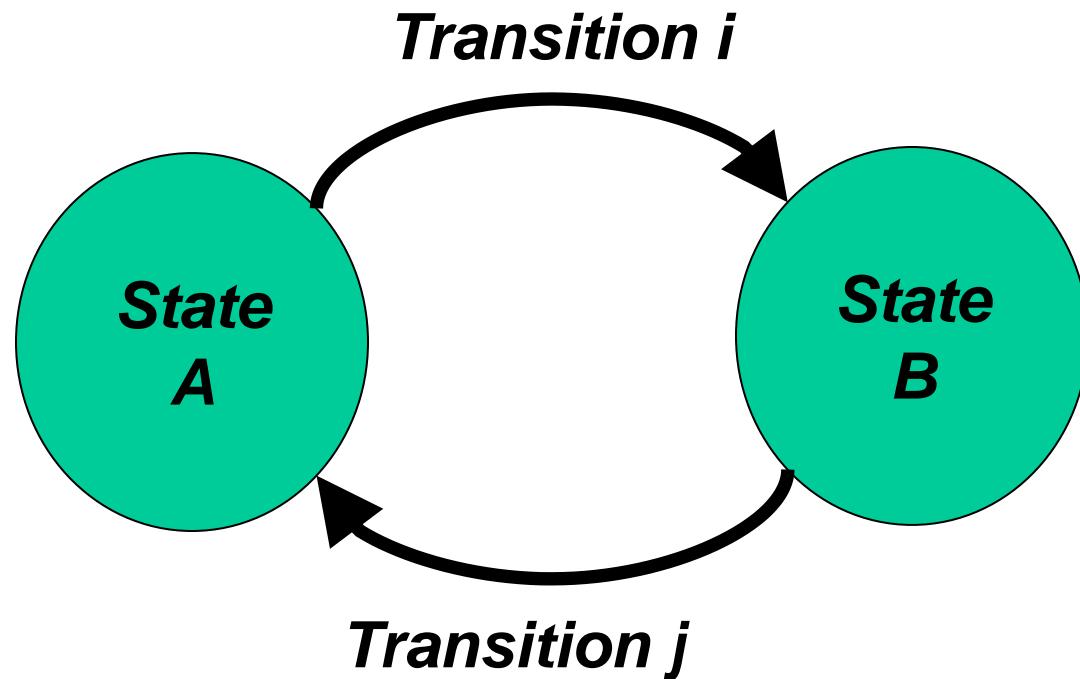
# State Models

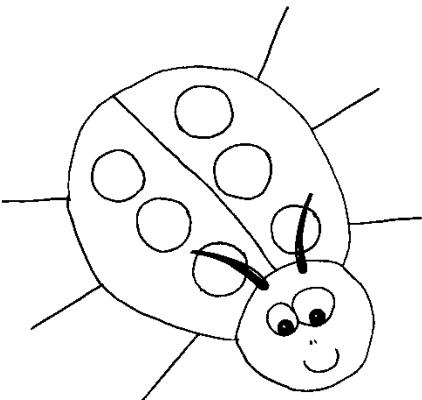
## “Stateful” Systems

- Transactions
- Embedded Systems
- Process
- Workflow
- User Interface



# State Models

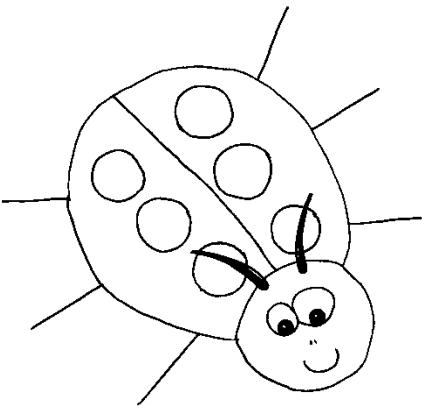




# State Models

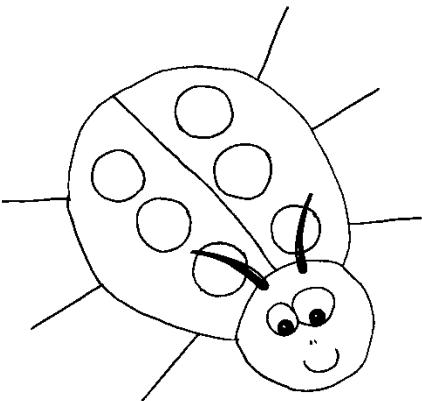
## Construction

1. Identify: States
2. Identify: Transitions
3. Identify: Triggers and Outcomes
4. Test: *Get to Each State*
5. Test: *Exercise Each Transition*
6. Test: *Cover Each Path*



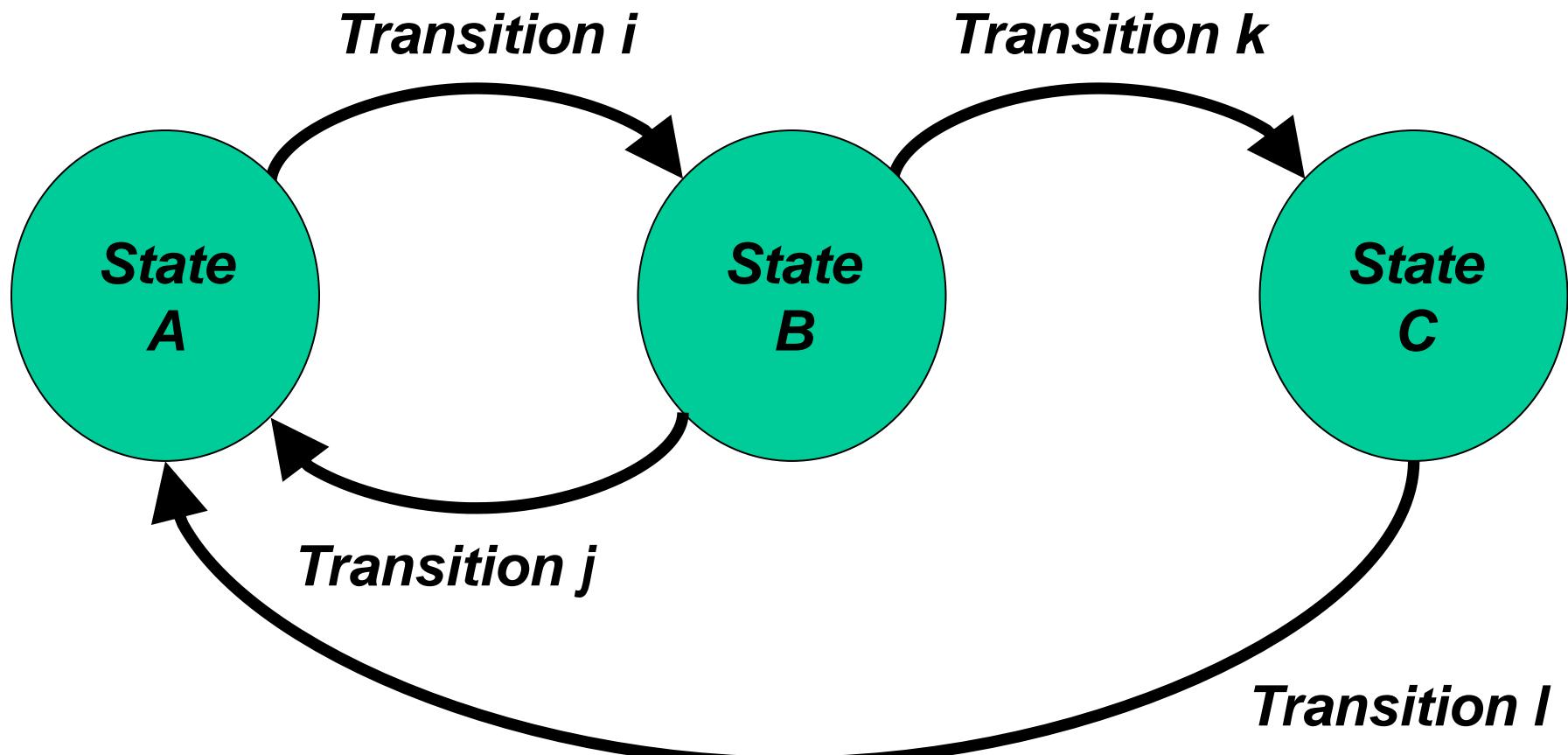
# San Diego Singles Pattern

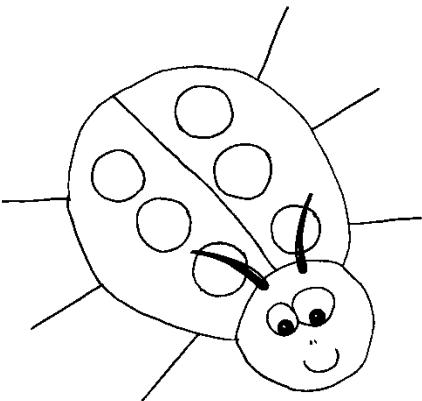
*State Models*



# State Models

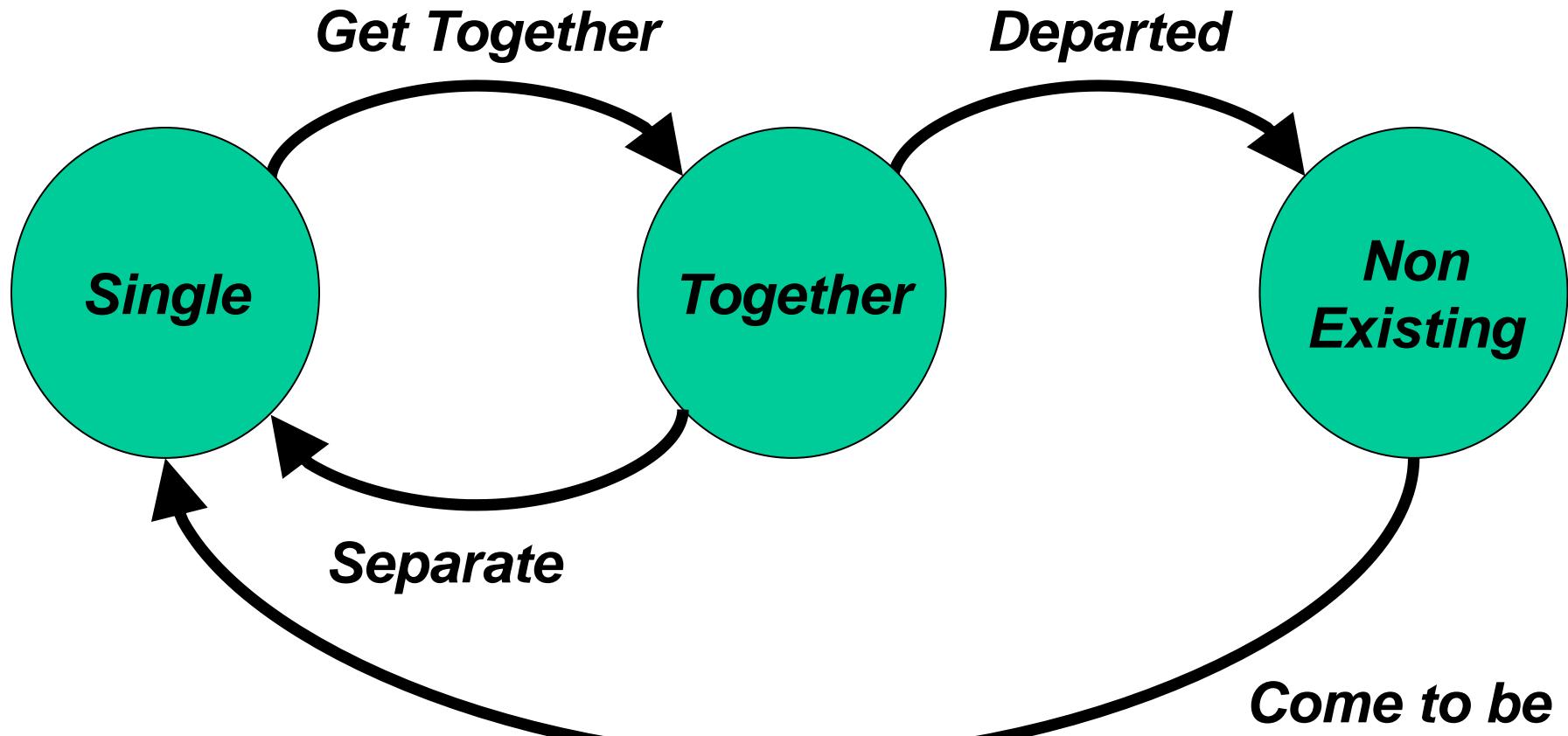
## San Diego Singles Pattern

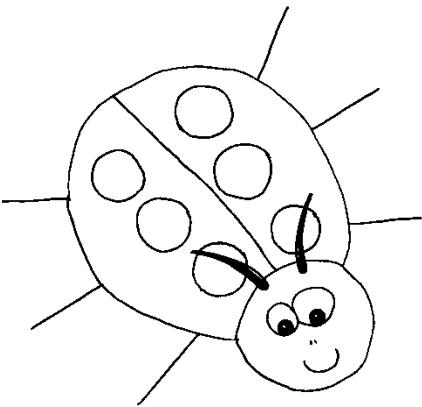




# State Models

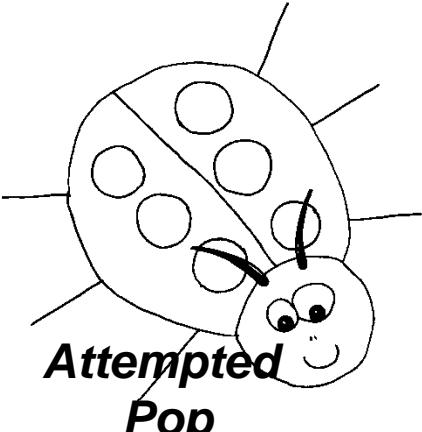
## San Diego Singles Pattern



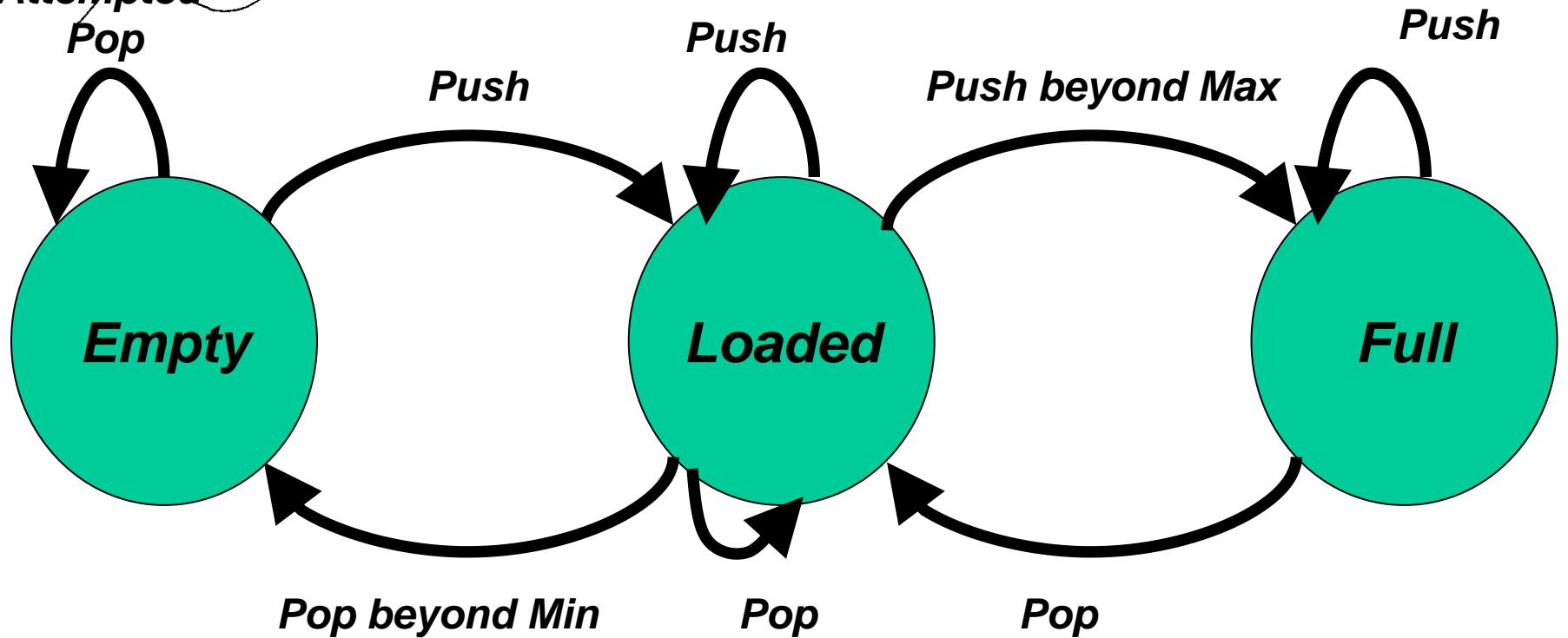


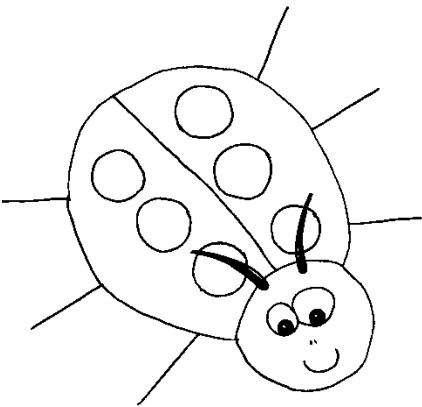
# Stack Pattern

*State Models*



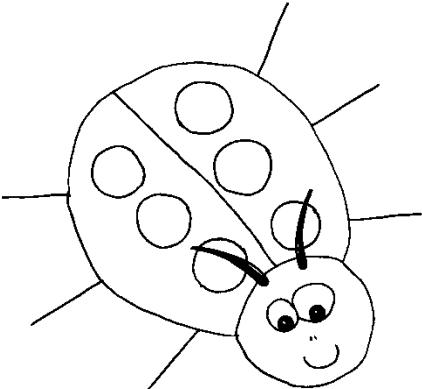
# State Models Stack Pattern





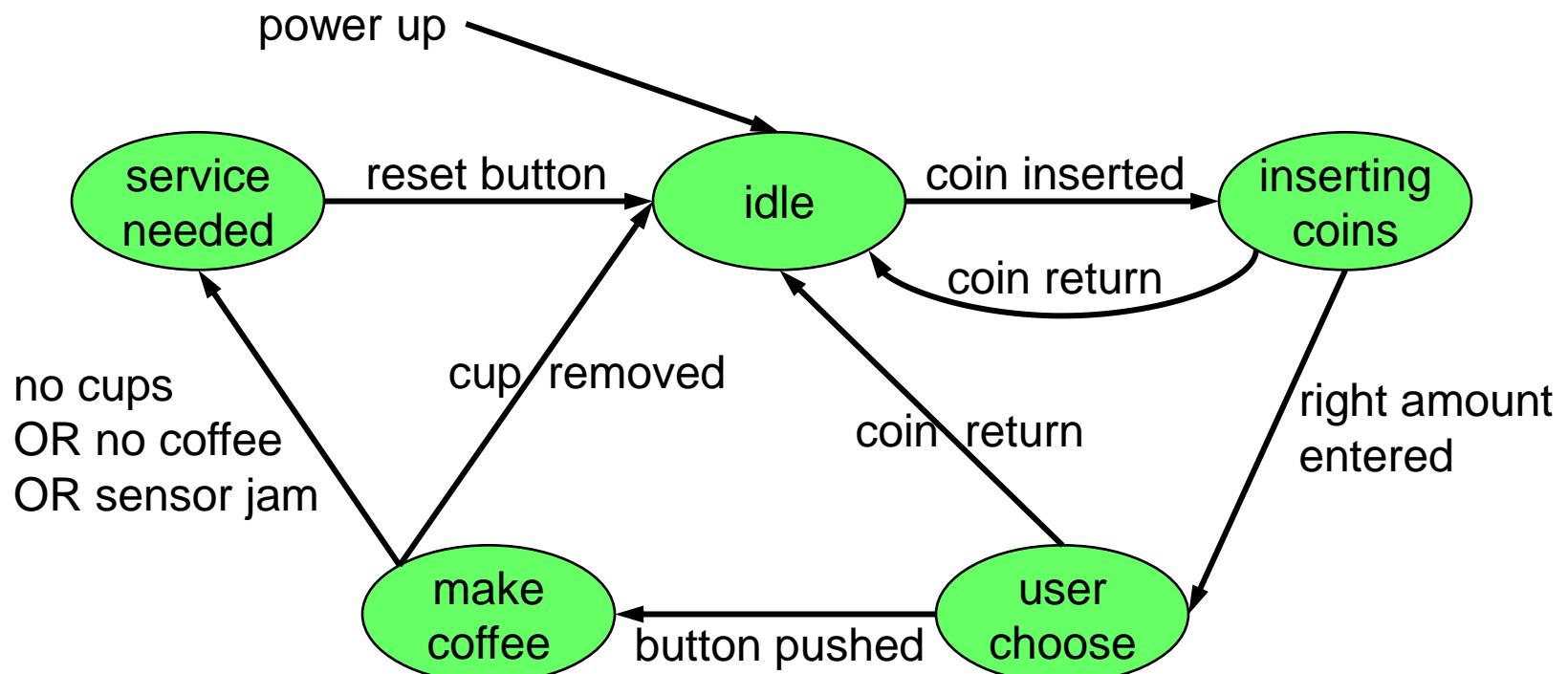
# Embedded Systems

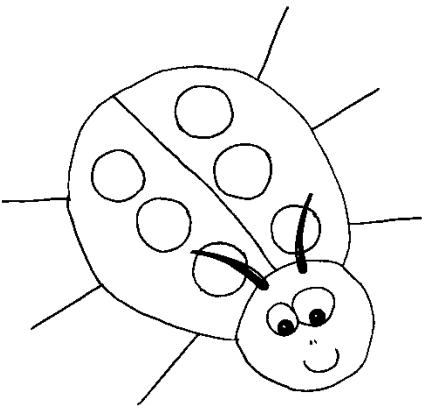
## *State Models*



# State Models

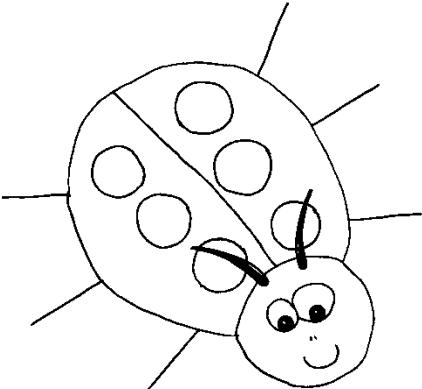
## Coffee Machine





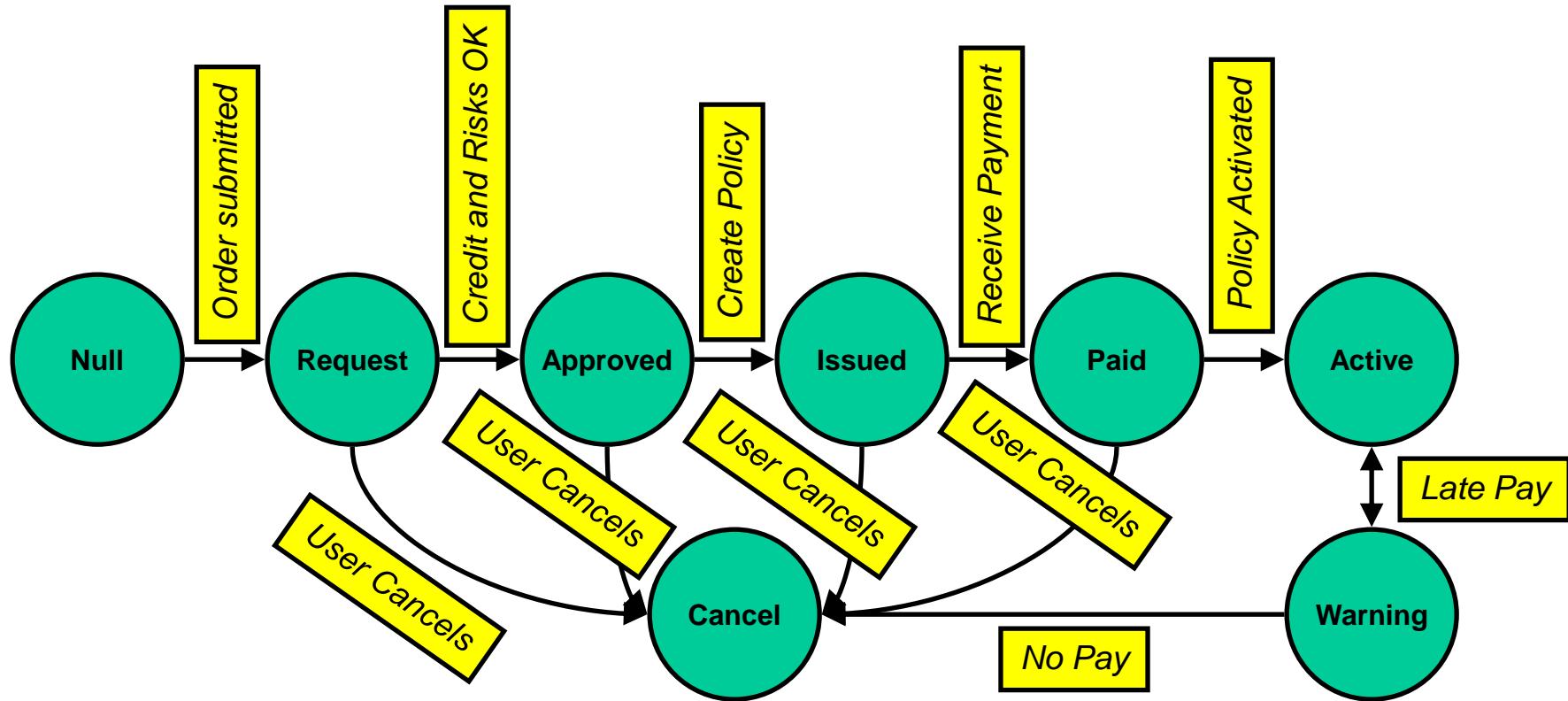
# Insurance Transactions

## *State Models*



# State Models

## Transaction

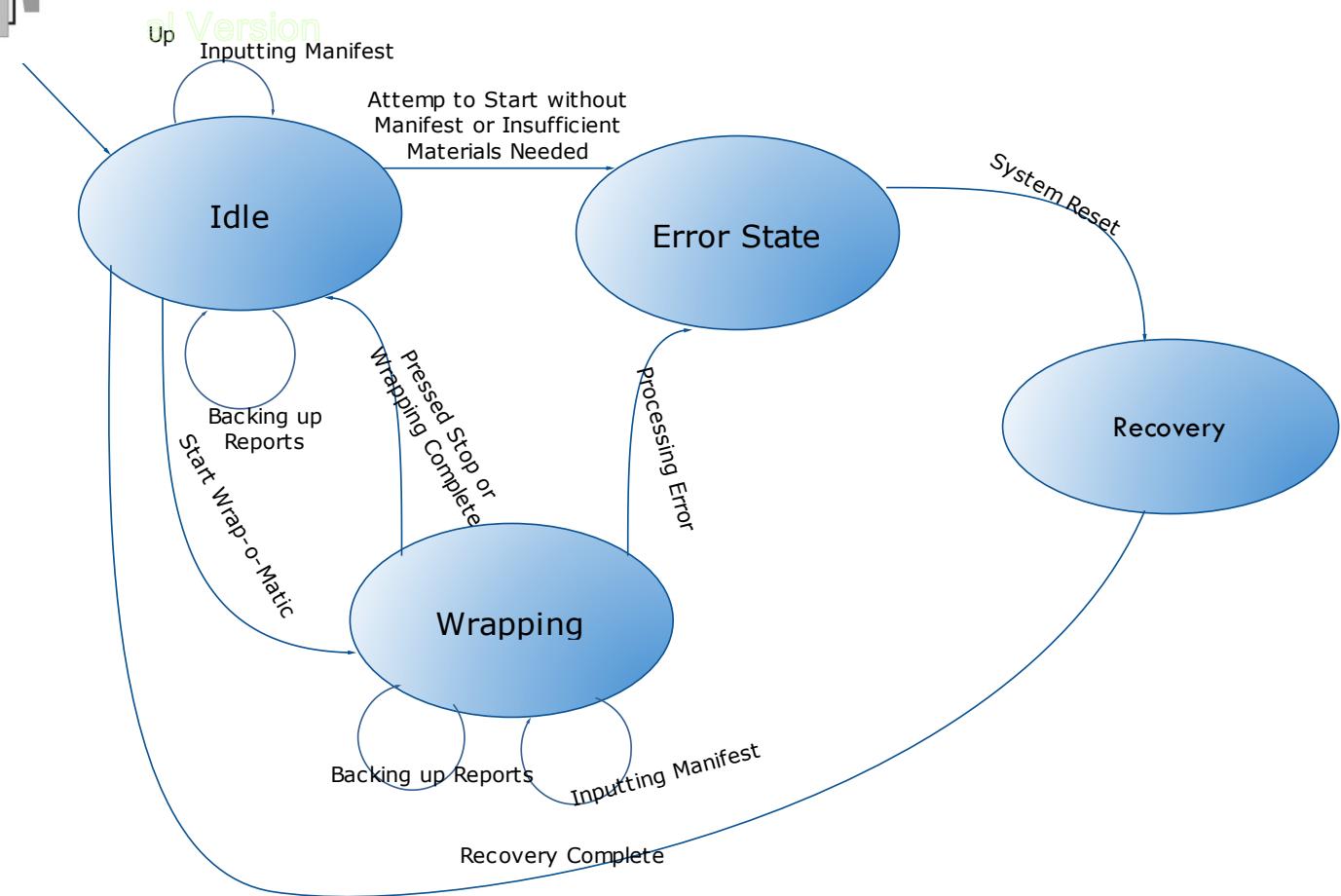


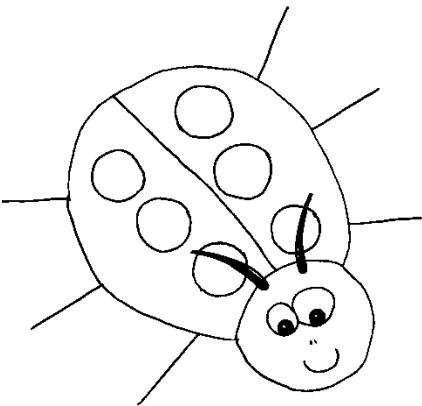


# Wrap-O-Matic

*State Models*

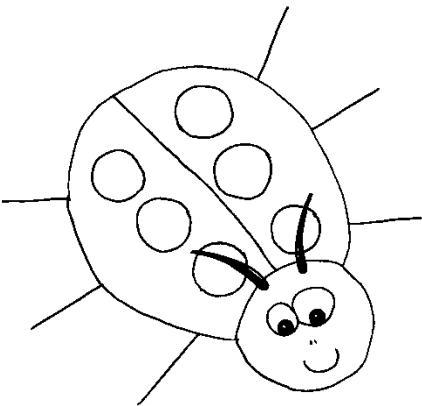
# State Models





# Test Design

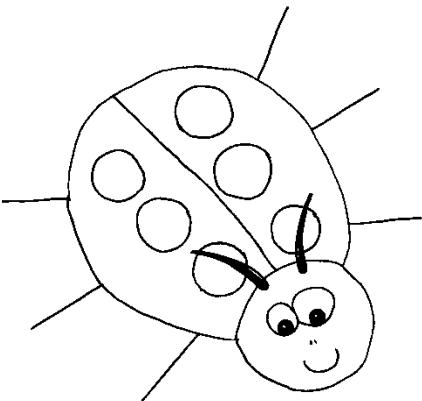
*Combination Testing*



# Pareto Principle

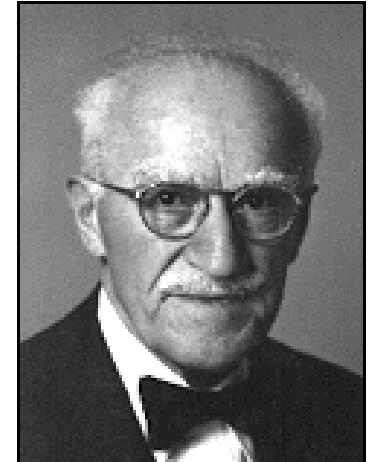
- Vilfredo Pareto, 1848 - 1923, Economist
  - 80% of the wealth was in the hands of 20% of the population

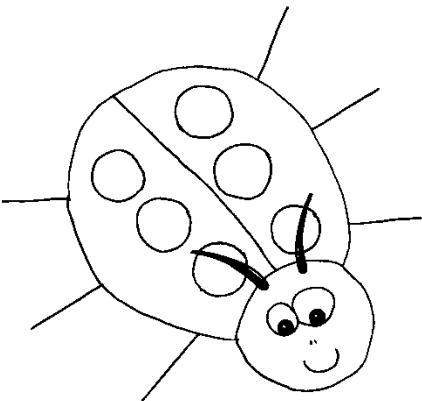




# Pareto Principle

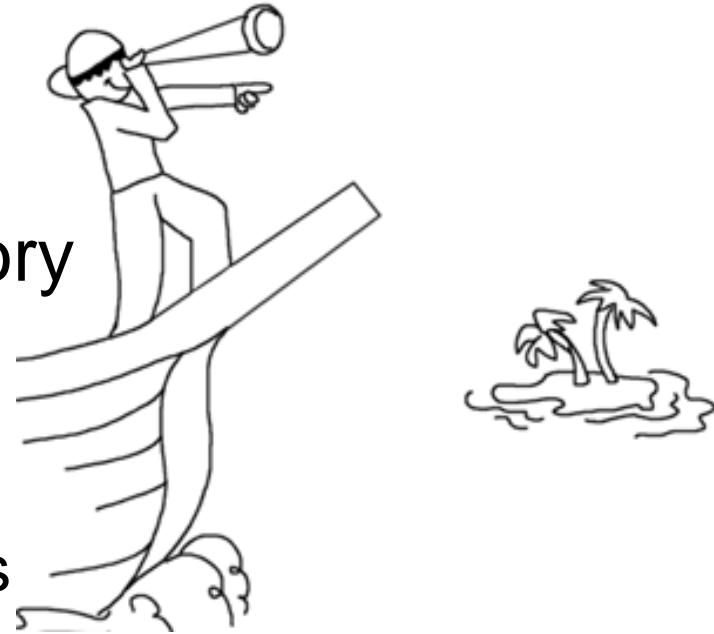
- Joseph Juran, 1903 - 2008,  
Quality Control Engineer
  - 1950 Quality Control Handbook
  - 20% of the study population accounts for  
80% of the measure under consideration
  - “ ... vital few and trivial many ... ”

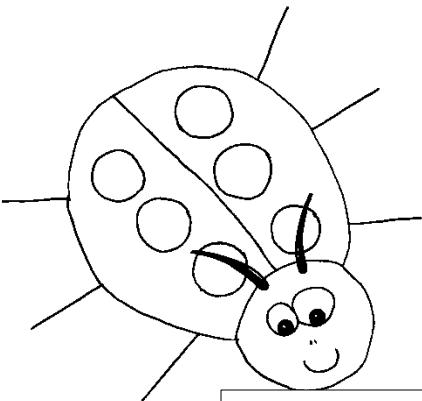




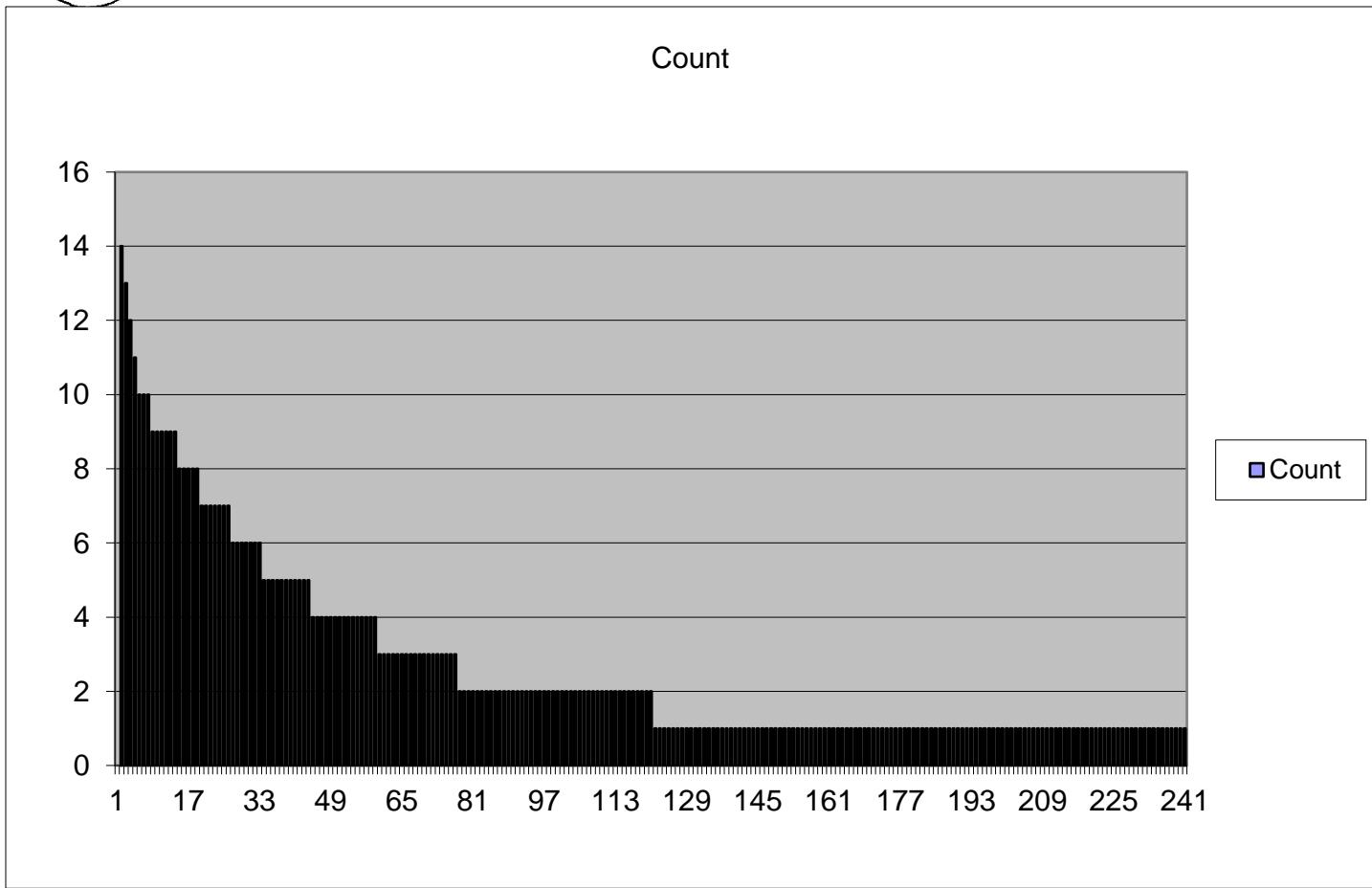
# Pareto Analysis

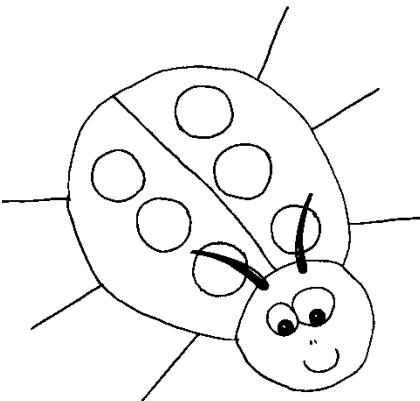
- Pareto Combinations
  - Start from transaction history
  - Create histogram
  - Identify sweet spot
    - 20% of the transaction types
    - 80% of the time





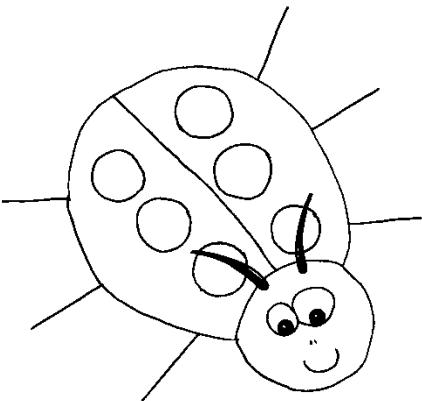
# Pareto Analysis

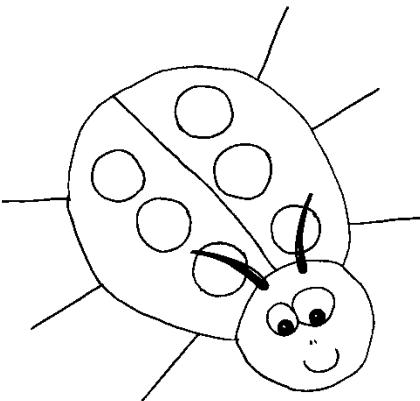




# Pairwise Combinations

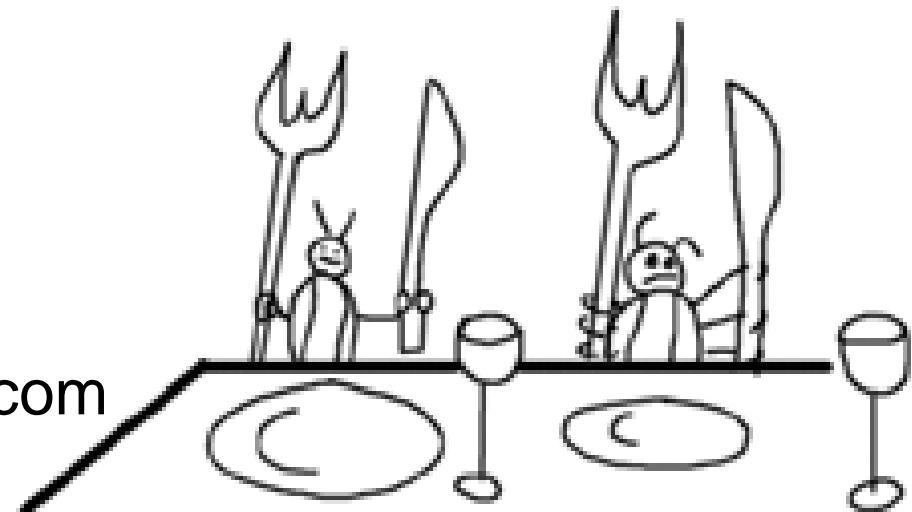
- Identify dependent variables of interest
- For each variable identify values (classes of special concern)
- For each pair of variables define at least one test case which exercise all possible combinations of values





# Pairwise Combinations

- Pairwise Combinations Test Tools
  - allpairs
    - [www.satisfice.com](http://www.satisfice.com)
  - pict
    - [www.microsoft.com](http://www.microsoft.com)
    - [www.amibugshare.com](http://www.amibugshare.com)





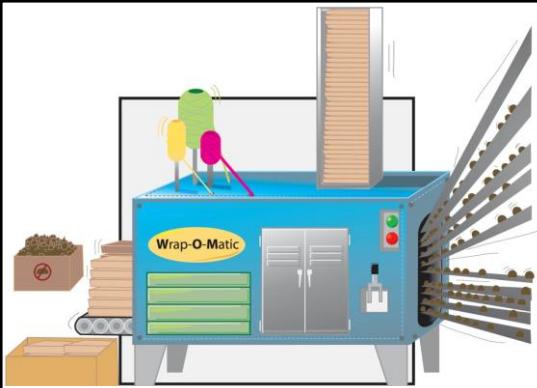
# Wrap-O-Matic

*Combinations*



# Pairwise Combinations

```
#  
# This is a sample model for testing volume create/delete functions  
#  
Choc:           White, Dark, Milk, Bitter, Semi, Swiss, Belg, Fudge  
Config:         Truffle, Bar, Turtle, BonBon, Praline, Filled  
Paper:          Wax, Foil, Paper, Tissue, None  
Ribbon:         Thread, Cord, Wide, Wire, None  
ConvSp:          Slow, Med, Fast, VeryFast  
BoxType:        Heart, Rect, Circle, Bag  
Visc:            Solid, Jelly, SemiSolid, Hollow  
BoxSiz:          Small, Med, Large  
Weight:          TooLight, InRange, TooHeavy  
Size:            One, Two, Three
```



# Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
1	Swiss	Filled	Tissue	Thread	Med	Rect	Hollow	Small	TooLight	Two
2	Swiss	Bar	None	Wire	Slow	Heart	Jelly	Med	TooHeavy	Three
3	Belg	Turtle	Tissue	Cord	Fast	Bag	Solid	Large	InRange	One
4	Dark	Bar	Wax	None	VeryFast	Circle	SemiSolid	Large	TooLight	One
5	Belg	Truffle	Paper	Wide	VeryFast	Heart	SemiSolid	Small	TooHeavy	Two
6	Swiss	Truffle	Foil	Wide	Fast	Circle	Hollow	Med	InRange	One
7	Belg	Bar	Foil	None	Med	Rect	Jelly	Small	InRange	Three
8	Semi	BonBon	Paper	Thread	VeryFast	Bag	Solid	Med	TooLight	Three
9	Bitter	Bar	Wax	Thread	Slow	Heart	Solid	Large	InRange	Two
10	Semi	Praline	Wax	Cord	Med	Circle	Jelly	Small	TooHeavy	One
11	Fudge	Filled	Wax	Wire	Fast	Bag	Hollow	Large	TooHeavy	Three
12	Semi	Filled	None	Cord	Slow	Rect	SemiSolid	Med	InRange	Two
13	Semi	Turtle	Foil	None	Fast	Heart	Jelly	Large	TooLight	Two
14	Bitter	Praline	Tissue	Wide	Med	Bag	SemiSolid	Med	TooLight	Three
15	Fudge	Truffle	None	Wire	Slow	Rect	Solid	Small	TooLight	One



# Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
16	White	BonBon	Foil	Wire	VeryFast	Rect	Hollow	Large	TooHeavy	Two
17	Dark	Filled	Paper	None	Med	Heart	Solid	Small	TooHeavy	One
18	White	Praline	Paper	Wide	Slow	Circle	Solid	Large	InRange	Three
19	Milk	Filled	Tissue	None	VeryFast	Bag	Jelly	Med	TooHeavy	Two
20	Dark	BonBon	None	Thread	Fast	Circle	Jelly	Small	InRange	Two
21	White	Bar	None	Cord	Slow	Bag	Hollow	Med	TooLight	One
22	Bitter	Turtle	Paper	Thread	VeryFast	Rect	Hollow	Small	TooHeavy	One
23	Belg	BonBon	Tissue	Wire	Slow	Circle	SemiSolid	Med	TooLight	One
24	Milk	BonBon	None	Cord	Med	Heart	SemiSolid	Large	InRange	Three
25	Swiss	Praline	Foil	Cord	VeryFast	Rect	SemiSolid	Large	InRange	Two
26	Milk	Bar	Paper	Wide	Fast	Rect	Hollow	Small	TooLight	One
27	Bitter	BonBon	Wax	Wide	Fast	Bag	Jelly	Small	InRange	Two
28	Belg	Filled	Wax	None	Slow	Heart	Hollow	Med	InRange	Two
29	Milk	Truffle	Foil	Thread	Slow	Bag	Jelly	Large	InRange	Three
30	Swiss	Turtle	Wax	Wire	Med	Bag	SemiSolid	Med	InRange	Three



# Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
31	Milk	Praline	None	Wire	Fast	Heart	Solid	Large	InRange	Three
32	Semi	Truffle	Wax	Wide	Med	Rect	Hollow	Small	InRange	One
33	Bitter	Truffle	Tissue	Cord	Fast	Circle	SemiSolid	Large	TooHeavy	Three
34	Fudge	Bar	Tissue	Thread	VeryFast	Heart	SemiSolid	Med	InRange	Two
35	Dark	Truffle	Foil	Wide	Slow	Bag	Hollow	Med	InRange	Three
36	Swiss	BonBon	None	None	VeryFast	Bag	Solid	Small	TooLight	Three
37	Fudge	BonBon	Paper	Cord	Med	Circle	Jelly	Small	InRange	Two
38	Fudge	Praline	Foil	None	Slow	Heart	Hollow	Small	InRange	One
39	White	Turtle	None	Wide	Fast	Circle	Jelly	Small	InRange	One
40	White	Truffle	Wax	Thread	Med	Heart	SemiSolid	Med	InRange	Three
41	White	Filled	Tissue	None	Slow	Circle	SemiSolid	Small	TooHeavy	Three
42	Fudge	Filled	Foil	Wide	Slow	Bag	Solid	Large	TooHeavy	One
43	Belg	Praline	None	Thread	Slow	Circle	Solid	Med	TooLight	One
44	Bitter	Turtle	Foil	None	Slow	Bag	SemiSolid	Small	TooLight	Three
45	Bitter	Filled	None	Wire	Slow	Heart	Jelly	Med	TooLight	One



# Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
46	Dark	Praline	Paper	Wire	Med	Rect	Hollow	Med	TooLight	One
47	Semi	Bar	Tissue	Wire	VeryFast	Rect	SemiSolid	Small	InRange	Three
48	Milk	Turtle	Wax	None	Med	Circle	SemiSolid	Large	InRange	Two
49	Dark	Turtle	Tissue	Cord	VeryFast	Bag	Hollow	Med	InRange	One
50	Swiss	Truffle	Paper	None	Slow	Bag	Jelly	Large	InRange	Two
51	Fudge	Turtle	Paper	None	Fast	Heart	Hollow	Large	InRange	Three



# Constrained Pairwise Combinations

```
#  
# This is a Wrap O Matic combinations testing example  
#  
Choc:           White, Dark, Milk, Bitter, Semi, Swiss, Belg, Fudge  
Config:         Truffle, Bar, Turtle, BonBon, Praline, Filled  
Paper:          Wax, Foil, Paper, Tissue, None  
Ribbon:         Thread, Cord, Wide, Wire, None  
ConvSp:          Slow, Med, Fast, VeryFast  
BoxType:        Heart, Rect, Circle, Bag  
Visc:            Solid, Jelly, SemiSolid, Hollow  
BoxSiz:          Small, Med, Large  
Weight:          Toolight, InRange, TooHeavy  
Size:            One, Two, Three  
#  
# No Wire on Hollow chocolates  
#  
IF [Ribbon] in {"Wire"} THEN [Visc] in {"Solid", "Jelly", "SemiSolid"};  
#  
# No Ribbon if No Paper  
#  
IF [Paper] in {"None"} THEN [Ribbon] in {"None"};
```

Wrapping Rules





# Constrained Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
1	Swiss	Filled	None	None	Slow	Circle	SemiSolid	Small	TooLight	Two
2	Fudge	Turtle	Tissue	Wide	VeryFast	Bag	Jelly	Large	InRange	One
3	Dark	Praline	Paper	Thread	Med	Heart	Solid	Med	TooHeavy	Three
4	Fudge	BonBon	Foil	Cord	Fast	Rect	Hollow	Large	TooHeavy	Two
5	Milk	Praline	Wax	Cord	VeryFast	Circle	Hollow	Med	TooLight	One
6	Belg	Praline	Wax	Wire	Fast	Heart	Jelly	Small	InRange	Two
7	Swiss	Praline	Foil	None	Med	Rect	SemiSolid	Large	InRange	Three
8	Bitter	Turtle	Foil	Wide	Fast	Bag	Solid	Small	TooLight	Three
9	Dark	BonBon	Paper	Thread	Slow	Bag	SemiSolid	Small	InRange	One
10	Semi	Bar	Tissue	Wire	VeryFast	Rect	Solid	Small	TooHeavy	One
11	Milk	BonBon	Paper	Wire	Med	Circle	Jelly	Large	TooLight	Three
12	Semi	BonBon	None	None	Slow	Bag	Solid	Med	InRange	Two
13	Fudge	Praline	Tissue	Wide	Slow	Heart	SemiSolid	Med	TooLight	Three
14	Fudge	Filled	Wax	None	Med	Heart	Solid	Large	TooHeavy	One
15	Belg	Bar	Foil	Thread	Slow	Rect	Hollow	Med	TooLight	Three



# Constrained Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
16	White	Bar	Tissue	Wide	Med	Circle	Hollow	Large	InRange	Two
17	Bitter	Truffle	Wax	Thread	Slow	Rect	Jelly	Large	TooHeavy	Two
18	Belg	Turtle	Paper	Cord	Fast	Circle	SemiSolid	Med	TooHeavy	One
19	White	Praline	Paper	Cord	VeryFast	Bag	Hollow	Small	TooHeavy	Three
20	Bitter	Bar	Foil	Cord	VeryFast	Heart	Jelly	Med	InRange	One
21	Dark	Bar	None	None	VeryFast	Rect	SemiSolid	Large	TooLight	Two
22	Bitter	BonBon	Wax	Wide	Med	Heart	Hollow	Small	TooHeavy	Three
23	Belg	Truffle	Foil	Wire	Med	Bag	Solid	Med	TooLight	Three
24	Milk	Turtle	None	None	Slow	Rect	Solid	Small	TooHeavy	Two
25	Milk	Filled	Tissue	Thread	Fast	Bag	Jelly	Med	InRange	Three
26	White	BonBon	None	None	Fast	Rect	Solid	Med	TooLight	One
27	Semi	Filled	Foil	Cord	Med	Circle	Hollow	Large	TooLight	Three
28	Milk	Bar	Wax	Wide	Fast	Bag	SemiSolid	Small	TooHeavy	Three
29	Semi	Turtle	Wax	Thread	Med	Heart	Jelly	Small	TooHeavy	Three
30	Swiss	Truffle	Tissue	None	VeryFast	Heart	Hollow	Med	InRange	One



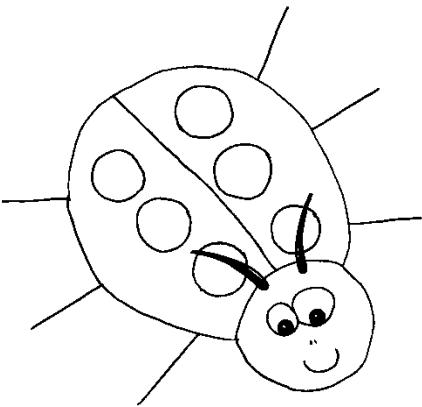
# Constrained Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
31	Swiss	BonBon	Paper	Thread	VeryFast	Bag	Jelly	Med	TooHeavy	Two
32	White	Truffle	Paper	Wire	Slow	Circle	SemiSolid	Small	TooHeavy	Three
33	Belg	Filled	Paper	Wide	VeryFast	Rect	Solid	Large	TooHeavy	Three
34	Fudge	Truffle	Paper	Cord	Fast	Circle	Solid	Small	TooLight	Three
35	White	Turtle	Wax	None	Slow	Heart	Jelly	Large	InRange	Two
36	Fudge	Bar	Tissue	Cord	Slow	Heart	Jelly	Med	TooHeavy	Three
37	Bitter	Praline	None	None	Med	Circle	Hollow	Med	InRange	Three
38	Fudge	Turtle	Tissue	Thread	Slow	Circle	Hollow	Small	TooLight	Two
39	Fudge	Truffle	None	None	VeryFast	Heart	Jelly	Med	InRange	One
40	Milk	Turtle	Foil	Wire	VeryFast	Heart	Jelly	Med	TooHeavy	Three
41	Swiss	Bar	Wax	Wide	Fast	Circle	Solid	Large	TooLight	Three
42	Swiss	Bar	Paper	Wire	Slow	Rect	SemiSolid	Large	TooHeavy	Three
43	Milk	Truffle	Wax	Wide	Fast	Circle	Solid	Small	TooHeavy	One
44	Dark	Filled	Tissue	Wire	Fast	Circle	Jelly	Small	InRange	Two
45	Semi	Praline	Paper	Wide	Fast	Bag	SemiSolid	Large	TooHeavy	One



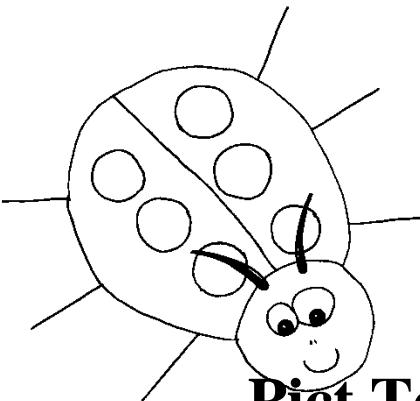
# Constrained Pairwise Combinations

ID	Choc	Config	Paper	Ribbon	ConvSp	BoxType	Visc	BoxSiz	Weight	Size
46	Bitter	Filled	Paper	Wire	Fast	Rect	SemiSolid	Large	TooLight	One
47	Swiss	Turtle	Foil	Cord	VeryFast	Rect	SemiSolid	Large	TooHeavy	Two
48	Belg	BonBon	Tissue	None	Slow	Rect	Jelly	Small	TooHeavy	Two
49	Belg	Truffle	None	None	Med	Bag	SemiSolid	Large	TooHeavy	Three
50	Dark	Truffle	Wax	Wide	Med	Rect	Hollow	Large	TooLight	Two
51	Fudge	Praline	Foil	Wire	Slow	Bag	Jelly	Small	TooLight	Three
52	Dark	Turtle	Foil	Cord	VeryFast	Rect	Hollow	Small	InRange	One
53	Bitter	Filled	Tissue	Cord	VeryFast	Bag	Jelly	Large	InRange	Three
54	White	Filled	Foil	Thread	Slow	Heart	Hollow	Med	InRange	Two
55	Semi	Truffle	Paper	None	Med	Heart	SemiSolid	Large	InRange	One



# Multiple Constraints

*Combinations*



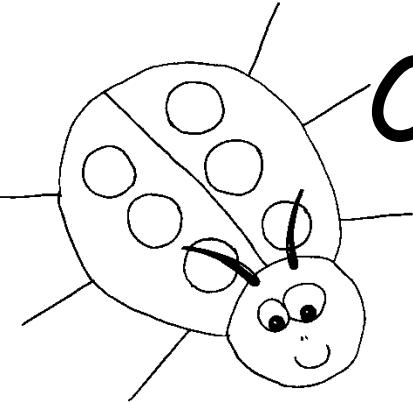
# Constrained Pairwise Combinations

## Pict Text Input (1)

```
#  
# Multiple RC case  
#  
BES:           Single, Multiple  
#  
RCs:           1, 2, 3, 4, 5  
#  
AS1:           On, Off, None  
Router1:       On, Off, None  
BBIM1:         On, Off, None  
MDS_CS1:       On, Off, None  
BAS1:          On, Off, None  
#  
AS2:           On, Off, None  
Router2:       On, Off, None  
BBIM2:         On, Off, None  
MDS_CS2:       On, Off, None  
BAS2:          On, Off, None
```

## Pict Text Input (2)

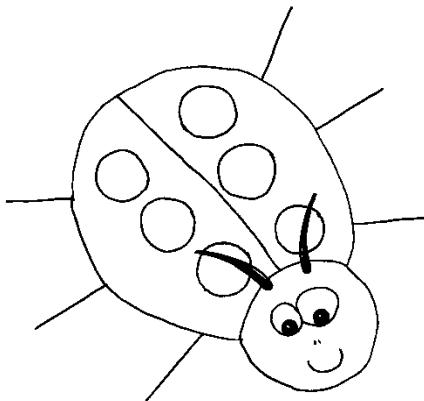
```
#  
AS3:           On, Off, None  
Router3:       On, Off, None  
BBIM3:         On, Off, None  
MDS_CS3:       On, Off, None  
BAS3:          On, Off, None  
#  
AS4:           On, Off, None  
Router4:       On, Off, None  
BBIM4:         On, Off, None  
MDS_CS4:       On, Off, None  
BAS4:          On, Off, None  
#  
AS5:           On, Off, None  
Router5:       On, Off, None  
BBIM5:         On, Off, None  
MDS_CS5:       On, Off, None  
BAS5:          On, Off, None
```



# Constrained Pairwise Combinations

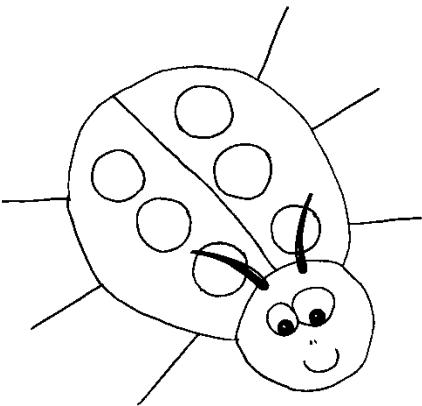
## Pict Text Input (3)

```
IF [RCs] < 5 THEN [AS5] in {"None"} ELSE [AS5] in {"On", "Off"} ;  
IF [RCs] < 5 THEN [Router5] in {"None"} ELSE [Router5] in {"On", "Off"} ;  
IF [RCs] < 5 THEN [BBIM5] in {"None"} ELSE [BBIM5] in {"On", "Off"} ;  
IF [RCs] < 5 THEN [MDS_CS5] in {"None"} ELSE [MDS_CS5] in {"On", "Off"} ;  
IF [RCs] < 5 THEN [BAS5] in {"None"} ELSE [BAS5] in {"On", "Off"} ;  
#  
IF [RCs] < 4 THEN [AS4] in {"None"} ELSE [AS4] in {"On", "Off"} ;  
IF [RCs] < 4 THEN [Router4] in {"None"} ELSE [Router4] in {"On", "Off"} ;  
IF [RCs] < 4 THEN [BBIM4] in {"None"} ELSE [BBIM4] in {"On", "Off"} ;  
IF [RCs] < 4 THEN [MDS_CS4] in {"None"} ELSE [MDS_CS4] in {"On", "Off"} ;  
IF [RCs] < 4 THEN [BAS4] in {"None"} ELSE [BAS4] in {"On", "Off"} ;  
#  
IF [RCs] < 3 THEN [AS3] in {"None"} ELSE [AS3] in {"On", "Off"} ;  
IF [RCs] < 3 THEN [Router3] in {"None"} ELSE [Router3] in {"On", "Off"} ;  
IF [RCs] < 3 THEN [BBIM3] in {"None"} ELSE [BBIM3] in {"On", "Off"} ;  
IF [RCs] < 3 THEN [MDS_CS3] in {"None"} ELSE [MDS_CS3] in {"On", "Off"} ;  
IF [RCs] < 3 THEN [BAS3] in {"None"} ELSE [BAS3] in {"On", "Off"} ;  
#  
IF [RCs] < 2 THEN [AS2] in {"None"} ELSE [AS2] in {"On", "Off"} ;  
IF [RCs] < 2 THEN [Router2] in {"None"} ELSE [Router2] in {"On", "Off"} ;  
IF [RCs] < 2 THEN [BBIM2] in {"None"} ELSE [BBIM2] in {"On", "Off"} ;  
IF [RCs] < 2 THEN [MDS_CS2] in {"None"} ELSE [MDS_CS2] in {"On", "Off"} ;  
IF [RCs] < 2 THEN [BAS2] in {"None"} ELSE [BAS2] in {"On", "Off"} ;
```



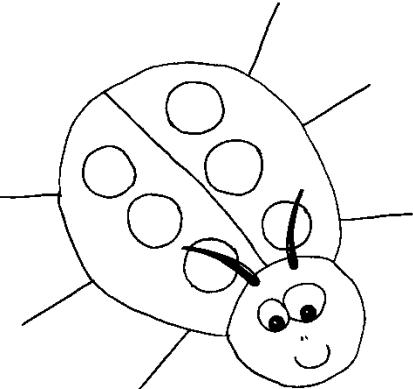
# Constrained Pairwise Combinations

ID	BES	RCs	AS1	Router1	BBIM1	MDS_CS1	BAS1	AS2	Router2	BBIM2	MDS_CS2	BAS2	AS3	Router3	BBIM3	MDS_CS3	BAS3	AS4	Router4	BBIM4	MDS_CS4	BAS4	AS5	Router5	BBIM5	MDS_CS5	BAS5
1	Mutliple	1	On	Off	None	On	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	
2	Single	5	Off	On	Off	On	On	Off	Off	Off	Off	On	On	Off	On	Off	Off	On	On	On	On	Off	Off	Off	On	Off	
3	Single	1	Off	None	On	Off	Off	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	
4	Mutliple	1	None	On	Off	None	On	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	
5	Single	2	Off	None	None	None	On	On	On	On	On	On	None	None	None	None	None	None	None	None	None	None	None	None	None	None	
6	Mutliple	3	On	On	None	Off	Off	Off	Off	Off	Off	Off	On	Off	On	None	None	None	None	None	None	None	None	None	None	None	
7	Mutliple	4	None	Off	Off	Off	Off	On	On	On	On	On	Off	On	Off	On	Off	Off	Off	Off	None	None	None	None	None	None	
8	Mutliple	4	On	Off	On	None	On	On	On	On	On	On	On	Off	On	Off	On	On	On	On	On	On	None	None	None	None	
9	Single	4	None	On	On	On	None	On	Off	Off	Off	On	Off	On	Off	On	Off	Off	On	Off	Off	None	None	None	None	None	
10	Single	3	On	None	Off	None	None	Off	Off	Off	On	On	On	Off	On	Off	None	None	None	None	None	None	None	None	None	None	
11	Single	5	On	None	None	None	On	Off	On	On	Off	Off	Off	On	Off	On	On	On	Off	Off	Off	On	On	On	Off	On	
12	Mutliple	5	None	None	On	On	None	On	On	Off	On	On	Off	On	On	On	On	On	Off	Off	Off	On	Off	Off	On	On	
13	Mutliple	2	None	Off	On	On	Off	Off	Off	Off	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	
14	Mutliple	5	Off	Off	None	Off	Off	On	Off	On	On	On	Off	Off	Off	Off	Off	Off	On	On	On	Off	On	Off	Off	On	
15	Single	3	Off	On	Off	On	Off	On	On	Off	On	On	Off	On	On	On	None	None	None	None	None	None	None	None	None	None	
16	Mutliple	5	None	On	None	Off	Off	On	Off	Off	On	On	On	On	Off	Off	Off	Off	Off	On	Off	On	On	On	Off	Off	
17	Single	5	Off	Off	On	None	Off	Off	On	On	On	Off	On	On	Off	On	On	Off	On	Off	Off	On	Off	Off	Off	Off	
18	Single	5	On	None	Off	Off	On	Off	Off	Off	Off	Off	Off	On	Off	On	Off	On	On	Off	Off	Off	On	Off	Off	Off	
19	Single	5	On	On	Off	Off	None	Off	On	On	Off	Off	On	Off	Off	On	On	Off	On	On	On	On	On	Off	On	On	
20	Single	4	Off	None	On	On	Off	Off	On	On	On	Off	On	Off	Off	On	Off	Off	Off	Off	None	None	None	None	None	None	
21	Mutliple	4	None	None	None	On	On	On	On	Off	On	Off	On	Off	On	On	Off	On	On	Off	Off	None	None	None	None	None	
22	Single	2	On	On	Off	Off	On	On	Off	On	Off	On	None	None	None	None	None	None	None	None	None	None	None	None	None	None	
23	Mutliple	5	None	On	On	On	None	Off	On	On	Off	On	On	Off	On	Off	On	On	On	On	On	On	On	Off	Off	Off	
24	Mutliple	5	None	None	None	None	On	Off	Off	Off	Off	On	On	Off	On	On	On	Off	On	Off	Off	Off	Off	On	Off	Off	
25	Single	3	None	Off	On	None	On	On	Off	Off	Off	On	Off	Off	Off	On	Off	None	None	None	None	None	None	None	None	None	
26	Mutliple	5	Off	Off	None	Off	None	Off	On	On	On	On	On	Off	Off	Off	Off	Off	Off	Off	On	Off	Off	On	On	Off	



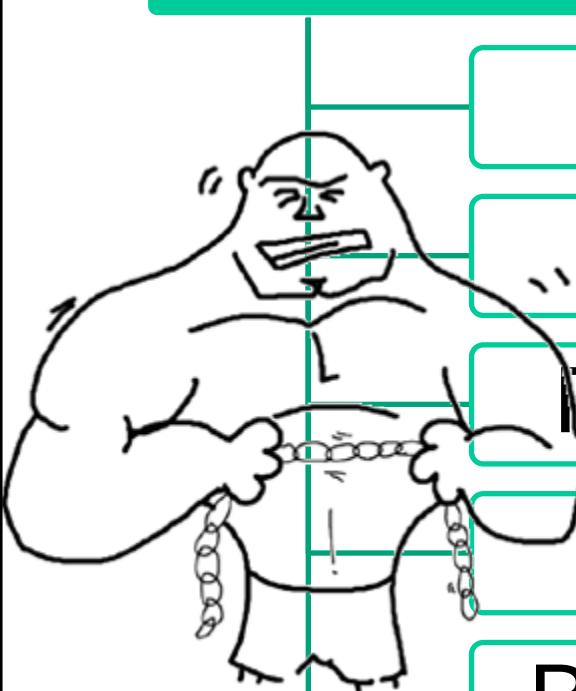
# Test Design

*Failure Mode Analysis*



# Failure Mode Analysis

## Failure Mode and Effects Analysis



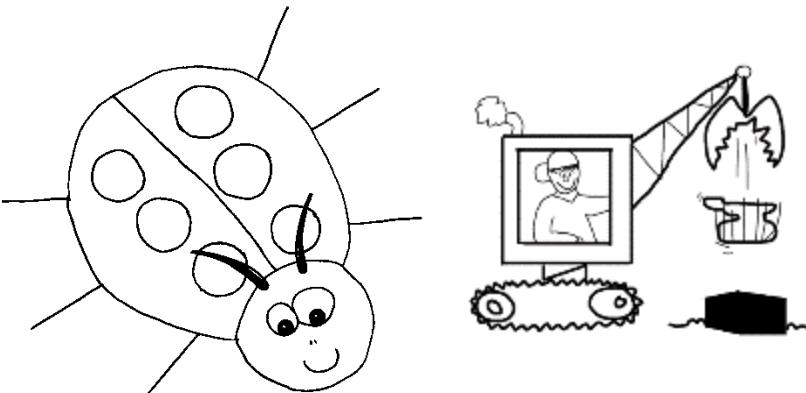
Create block diagram

List classes of users

Find elements which can fail

Create usage/failure table

Rank potential failure impacts



# Failure Mode Analysis

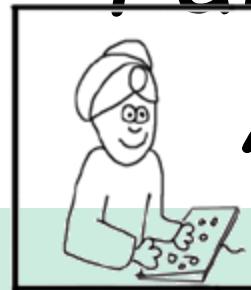
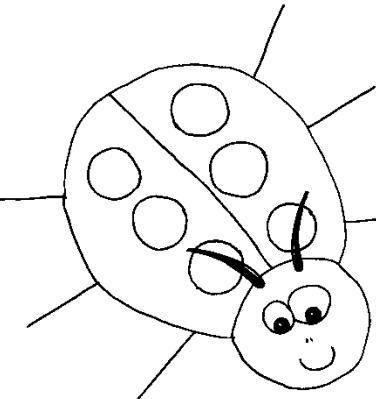
## Create block diagram

### Blocks

- System Elements
- Processes
- Subsystems

### Connector

- Logical relationships
- Physical relationships
- Process invocations



# Failure Mode Analysis

## List classes of users

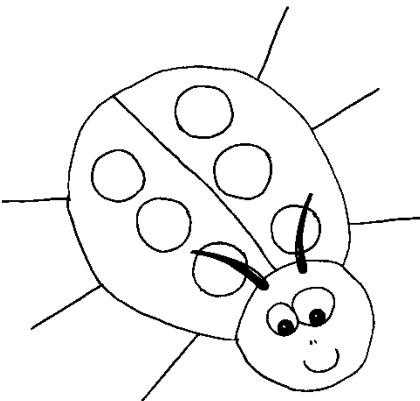
Who are the users?

What are they doing?

Classes

Personas

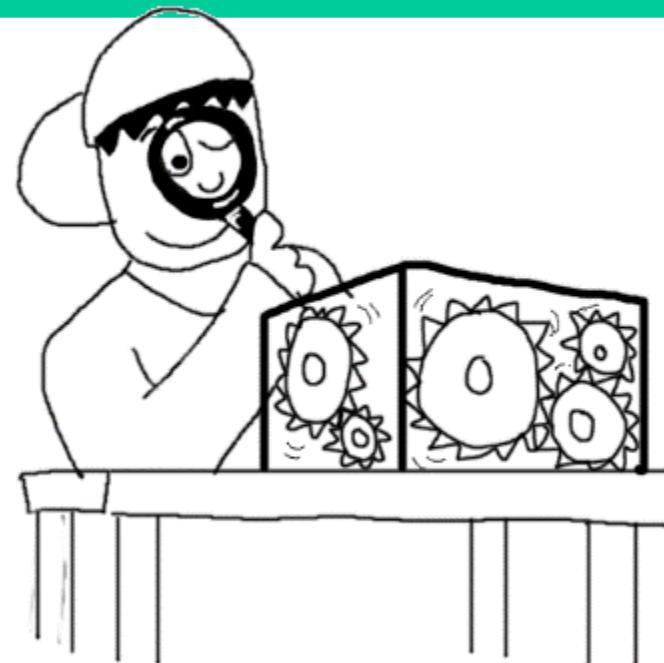
Context

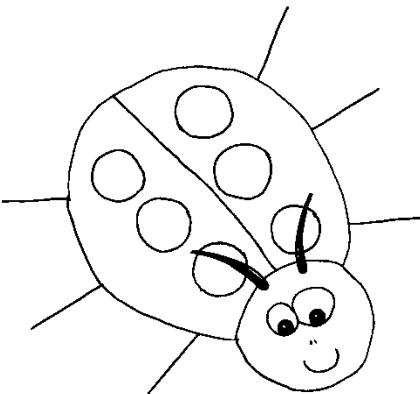


# Failure Mode Analysis

Find elements which can fail

- Break
- Constrained
- Slow
- ABEND
- Crash



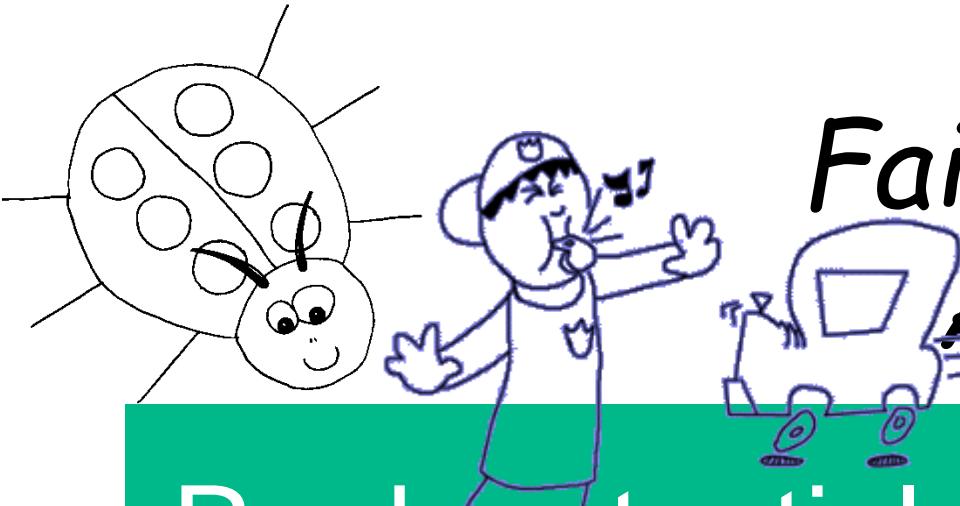


# Failure Mode Analysis

Create usage/failure table

- Scenarios
- Interesting blocks





# Failure Mode Analysis

## Rank potential failure impacts

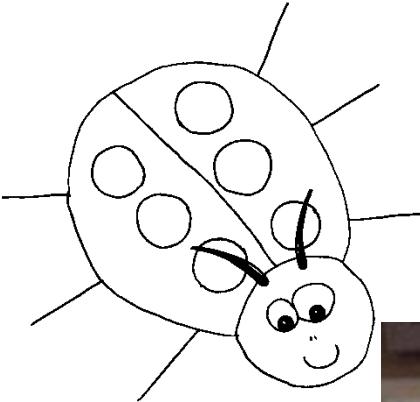
S4 – Cosmetic <ul style="list-style-type: none"><li>• User can accomplish the task with some inconvenience</li></ul>	S3 – Work around <ul style="list-style-type: none"><li>• User can accomplish task by working around the problem</li></ul>	S2 – No Work around <ul style="list-style-type: none"><li>• User cannot accomplish some tasks</li></ul>	S1 – Show stopper <ul style="list-style-type: none"><li>• User cannot accomplish any tasks</li></ul>	S0 – No Impact <ul style="list-style-type: none"><li>• User can accomplish the task</li></ul>
--	---	---	--	---



# Wrap-O-Matic

*Failure Mode Analysis*

# Failure Modes



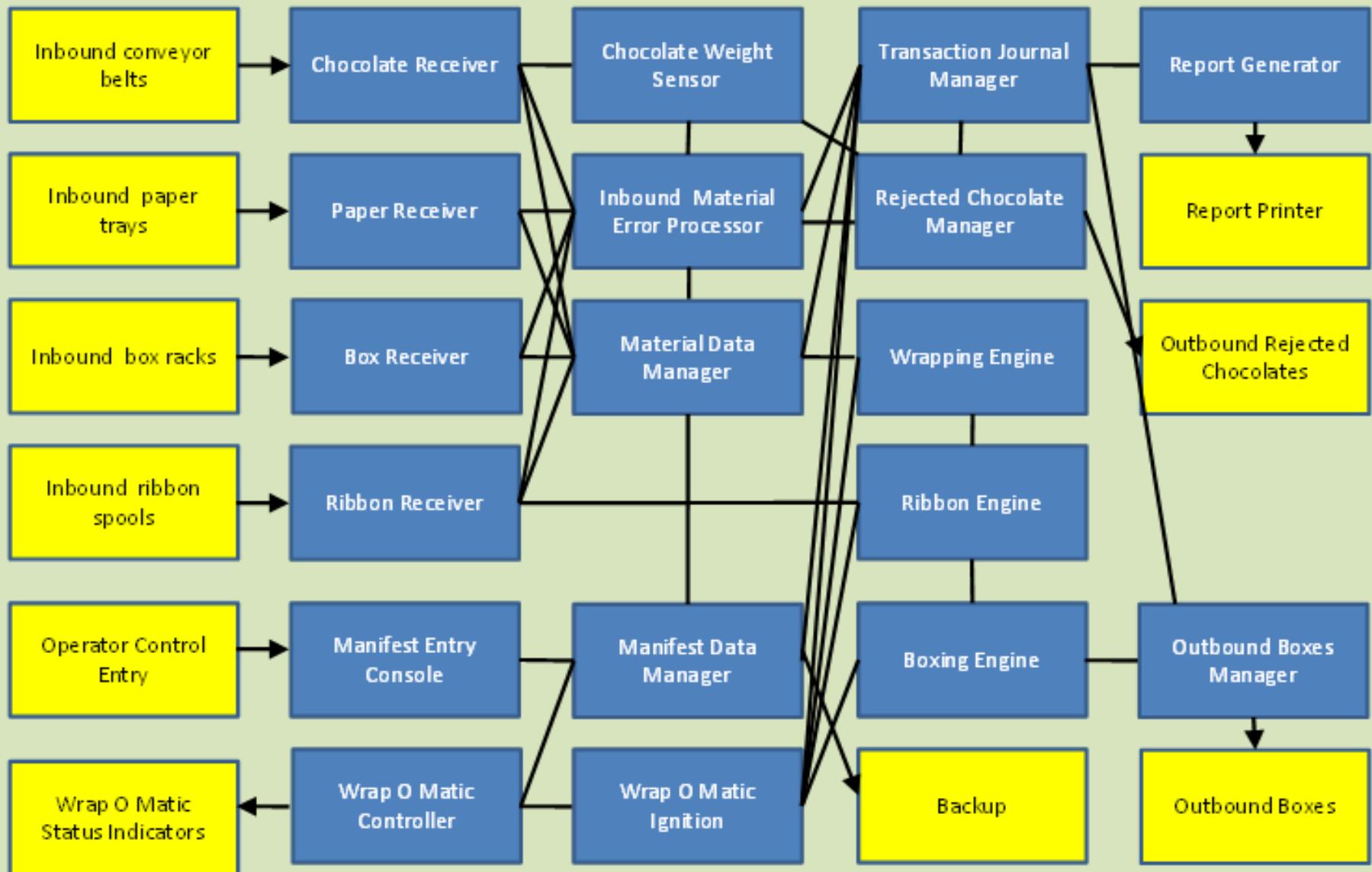
# Whiteboarding

The whiteboard contains two main sections:

- System Diagram:** A hand-drawn flowchart illustrating a process. It starts with a "Client" box on the left, which has an arrow pointing to a "LOGIC" box. From "LOGIC", arrows point to "INV MAN" and "SHIP PROC". "INV MAN" has an arrow pointing to "SHIP PROC". "SHIP PROC" has an arrow pointing to "Ext Shipment".
- Failure Mode Matrix:** A table titled "User Scenario" with columns for "DICK MAN", "DICK MAN", "INV MAN", and "SHIP PROC". The rows represent different user scenarios:

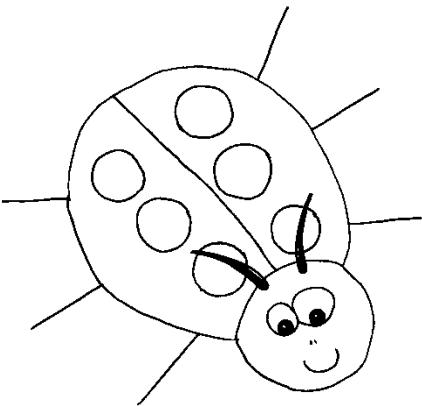
User	Scenario	DICK MAN	DICK MAN	INV MAN	SHIP PROC
Client	Place order	↑	↑	↑	↑
Client	Stat Order	↑	↑	↑	-
Client	Cancel order	↑	↑	↑	-
Client	Check order	↑	↑	↑	-
INV MAN	Update INV	-	-	↑	-
INV MAN	Resched	-	-	↑	-
Ext. Sys	Get Ship Rec	-	-	↑	↑
INV MAN	Background op	-	-	↑	-

**Date:** 05/11/2012



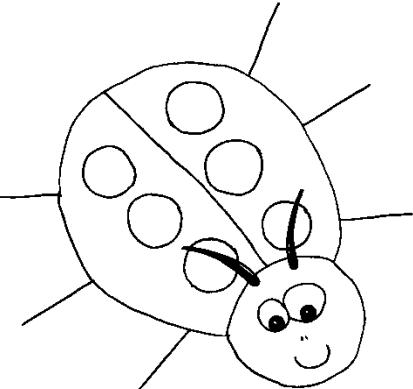
*Wrap O Matic Process Block Diagram*





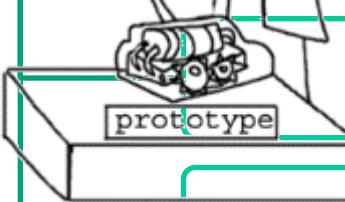
# Test Design

*Syntax Diagrams*



# Syntax Diagrams

Syntax validation



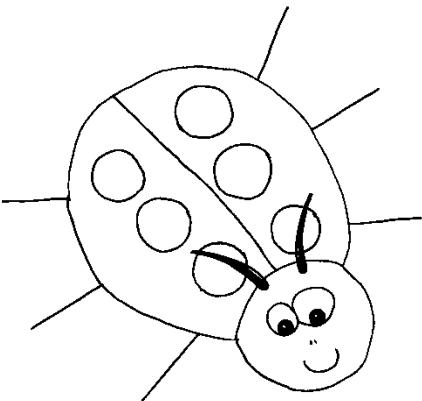
Parsers

Patterns

Format

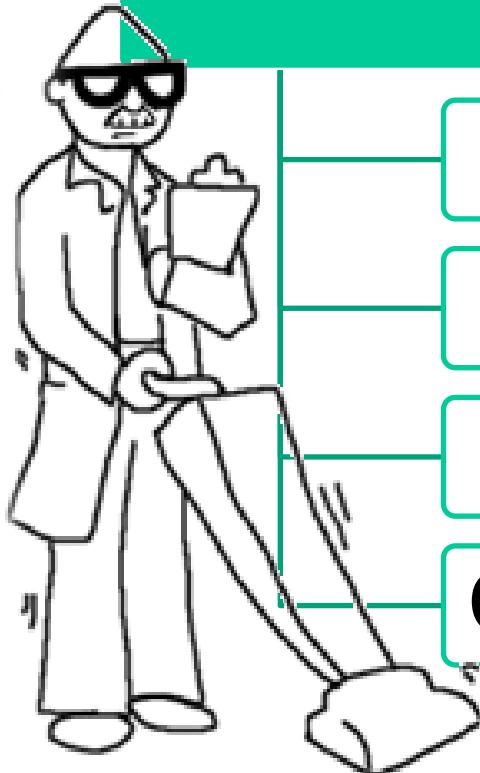
XML

Static Analysis



# Syntax Diagrams

## Syntax Flow Tests

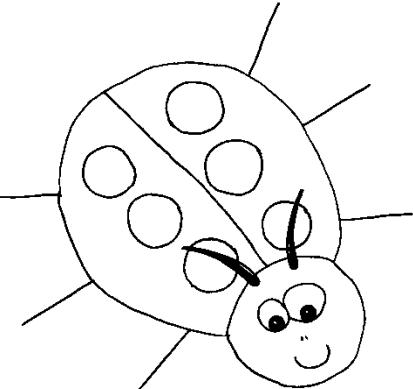


Identify syntax rules

Circles contain objects

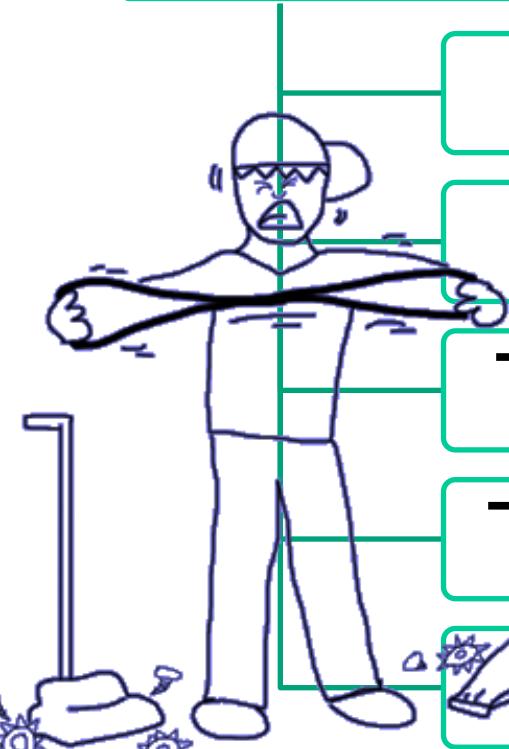
Arrows indicate order

Create syntax diagram (automata)



# Syntax Diagrams

Some interesting syntax tests



Exercises valid paths

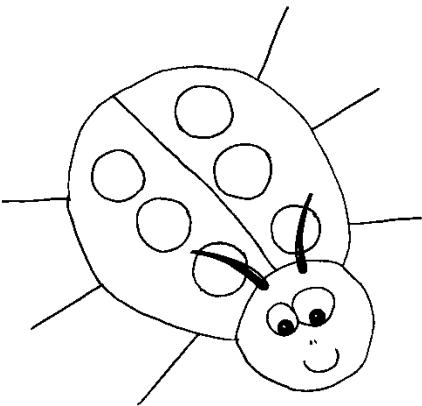
Attempt invalid paths

Try out of sequence operands

Try invalid/unexpected objects

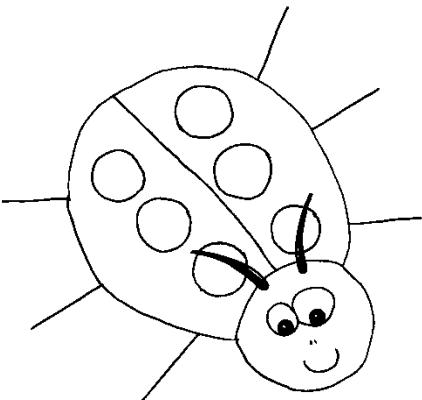


Multi-thread



# JSON Number Parsing

*Syntax Diagram*



# Syntax Diagrams

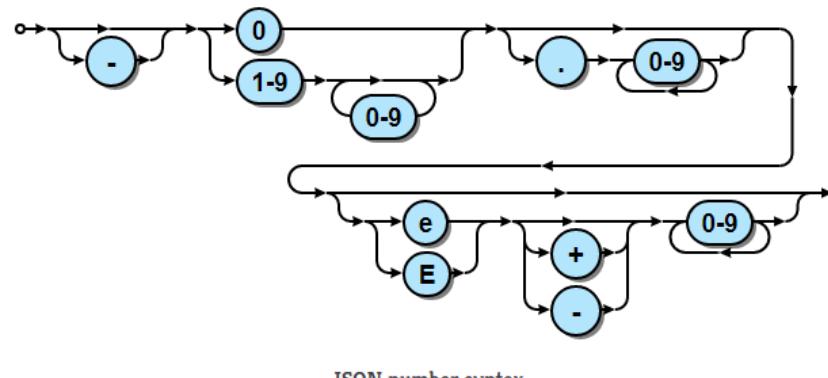
## Syntrax

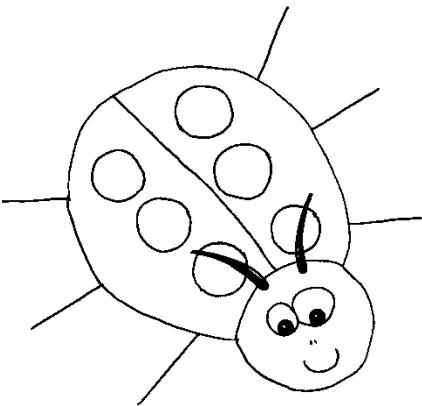
Syntrax is a railroad diagram generator. It creates a visual illustration of the grammar used for programming languages. A specification file describes the syntax as a hierarchy of basic elements. This is processed into an image representing the same syntax with interconnected nodes.

The specification is a set of nested Python function calls:

```
indentstack(10,
    line(opt('-', choice('0', line('1-9', loop(None, '0-9'))),
        opt('.', loop('0-9', None))),
    line(opt(choice('e', 'E'), choice(None, '+', '-'), loop('0-9', None)))
)
```

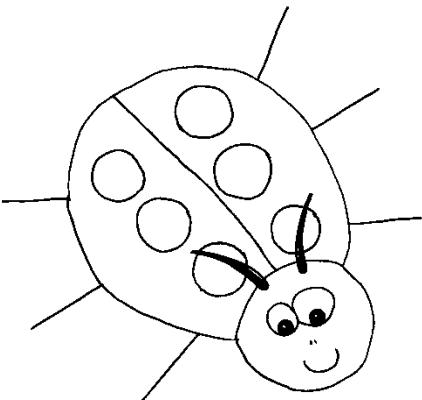
This is processed by Syntrax to generate an SVG image:





# Combinations and Permutations

*... With Python ...*

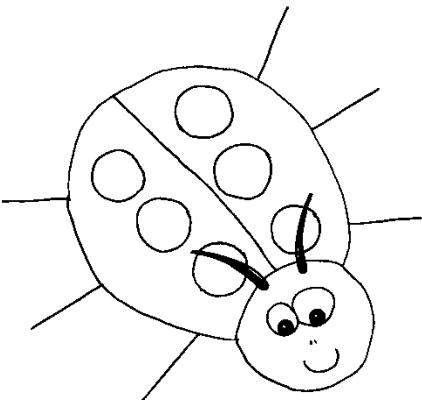


# Combinations & Permutations

Explore  
selections

Multiple  
variables

Combinations

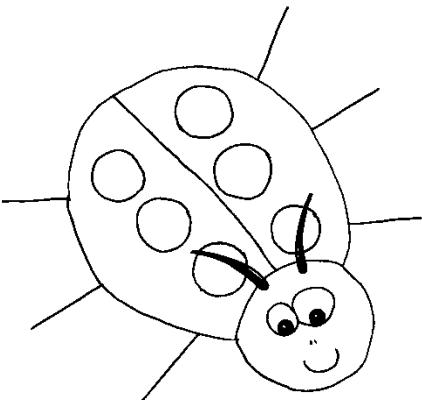


# Combinations & Permutations

Explore  
order

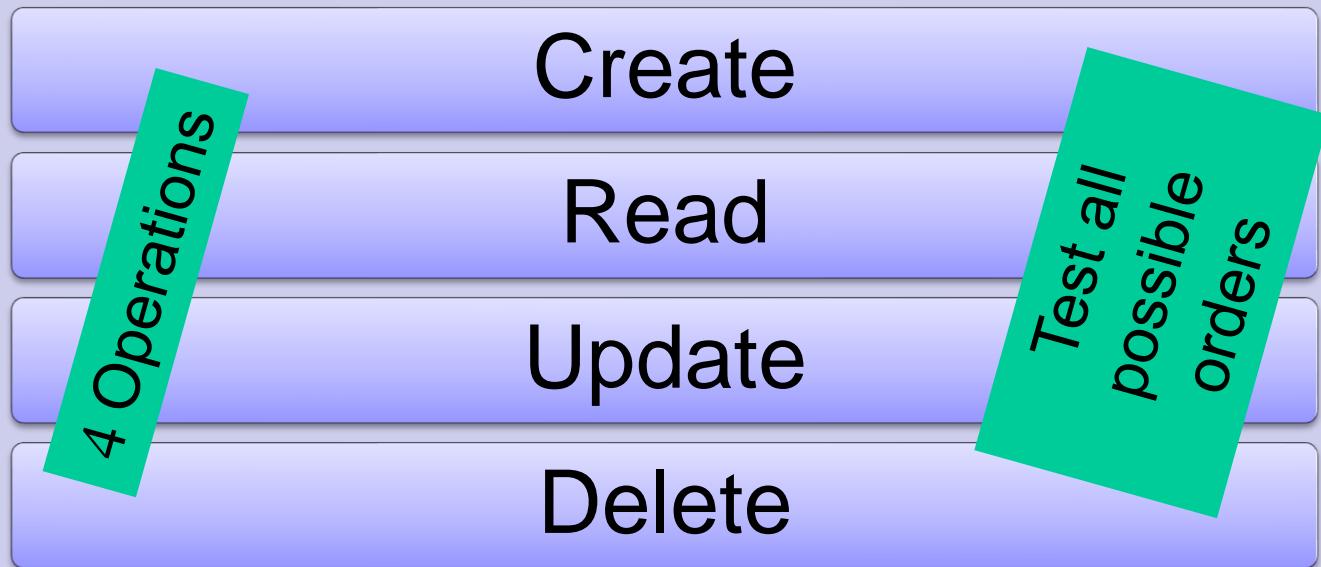
Multiple  
values

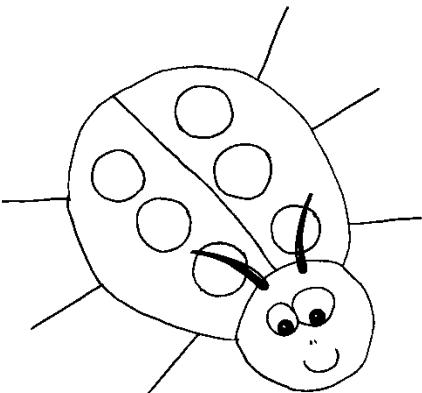
## Permutations



# Permutation Example

## Example Application

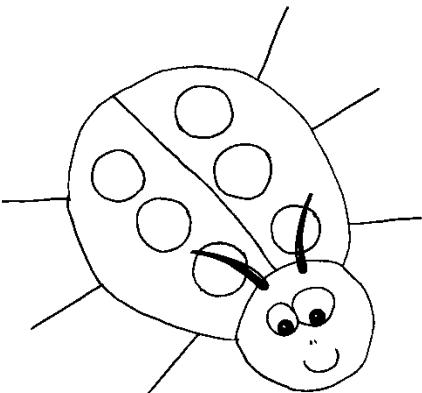




# Permutation Example

```
import itertools  
listA = ["Create","Read","Update","Delete"]  
perm = itertools.permutations(listA)  
for i in list(perm):  
    print(i)
```

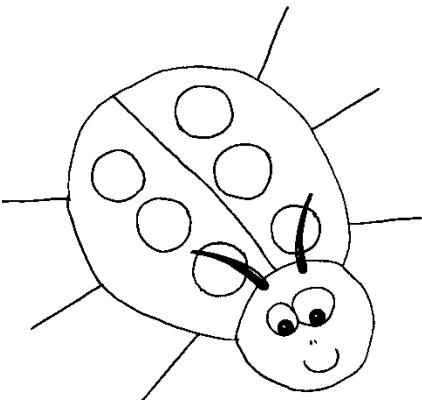
Python



# Permutation Example

```
('Create', 'Read', 'Update', 'Delete')
('Create', 'Read', 'Delete', 'Update')
('Create', 'Update', 'Read', 'Delete')
('Create', 'Update', 'Delete', 'Read')
('Create', 'Delete', 'Read', 'Update')
('Create', 'Delete', 'Update', 'Read')
('Read', 'Create', 'Update', 'Delete')
('Read', 'Create', 'Delete', 'Update')
('Read', 'Update', 'Create', 'Delete')
('Read', 'Update', 'Delete', 'Create')
('Read', 'Delete', 'Create', 'Update')
('Read', 'Delete', 'Update', 'Create')
('Update', 'Create', 'Read', 'Delete')
('Update', 'Create', 'Delete', 'Read')
('Update', 'Read', 'Create', 'Delete')
('Update', 'Read', 'Delete', 'Create')
('Update', 'Delete', 'Create', 'Read')
('Update', 'Delete', 'Read', 'Create')
('Delete', 'Create', 'Read', 'Update')
('Delete', 'Create', 'Update', 'Read')
('Delete', 'Read', 'Create', 'Update')
('Delete', 'Read', 'Update', 'Create')
('Delete', 'Update', 'Create', 'Read')
('Delete', 'Update', 'Read', 'Create')
```

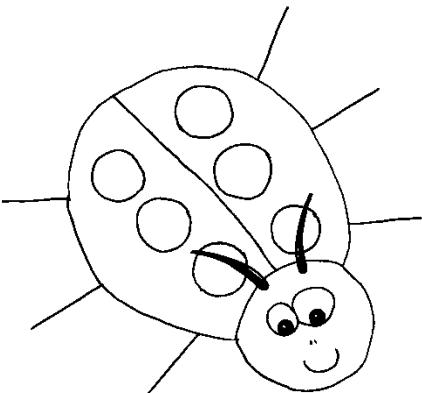
24  
Permutations



# Combination Example

## Example Application

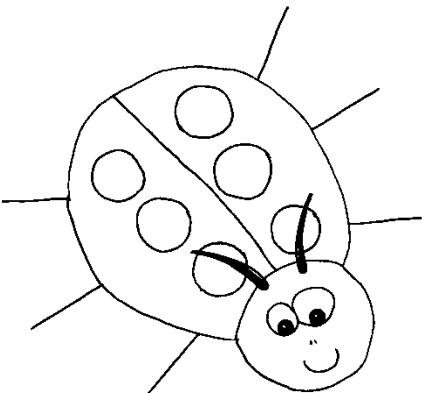




# Permutation Example

```
import itertools  
listA =  
["Create","Read","Update","Delete","Save","  
Restore","Reset"]  
perm = itertools.combinations(listA,3)  
for i in list(perm):  
    print(i)
```

Python

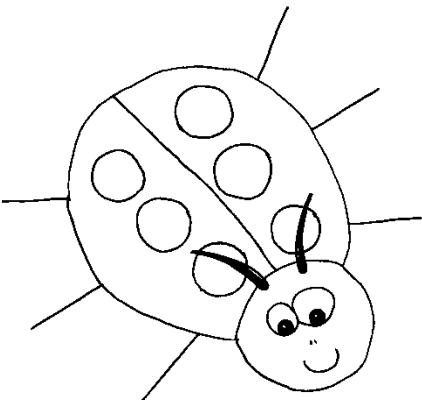


# Combination Example

```
('Create', 'Read', 'Update')
('Create', 'Read', 'Delete')
('Create', 'Read', 'Save')
('Create', 'Read', 'Restore')
('Create', 'Read', 'Reset')
('Create', 'Update', 'Delete')
('Create', 'Update', 'Save')
('Create', 'Update', 'Restore')
('Create', 'Update', 'Reset')
('Create', 'Delete', 'Save')
('Create', 'Delete', 'Restore')
('Create', 'Delete', 'Reset')
('Create', 'Save', 'Restore')
('Create', 'Save', 'Reset')
('Create', 'Restore', 'Reset')
('Read', 'Update', 'Delete')
('Read', 'Update', 'Save')
('Read', 'Update', 'Restore')
('Read', 'Update', 'Reset')
('Read', 'Delete', 'Save')
('Read', 'Delete', 'Restore')
('Read', 'Delete', 'Reset')
('Read', 'Save', 'Restore')
('Read', 'Save', 'Reset')
```

```
('Read', 'Restore', 'Reset')
('Update', 'Delete', 'Save')
('Update', 'Delete', 'Restore')
('Update', 'Delete', 'Reset')
('Update', 'Save', 'Restore')
('Update', 'Save', 'Reset')
('Update', 'Restore', 'Reset')
('Delete', 'Save', 'Restore')
('Delete', 'Save', 'Reset')
('Delete', 'Restore', 'Reset')
('Save', 'Restore', 'Reset')
```

35  
Combinations

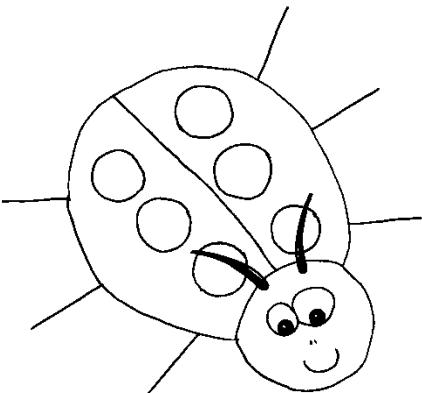


# Combination & Permutation Example

## Example Application

7 Operations	Create
	Read
	Update
	Delete
	Save
	Restore
	Reset

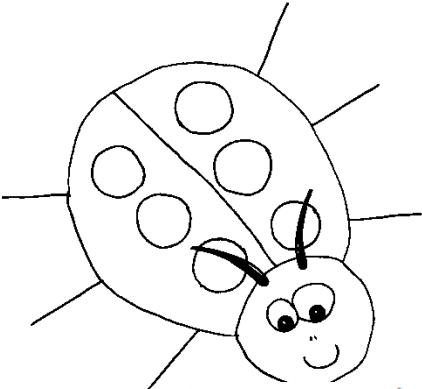
Try every permutation of every combination of 3



# Combination & Permutation Example

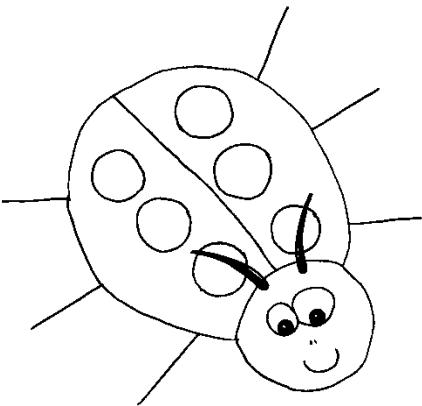
```
import itertools  
  
listA =  
["Create","Read","Update","Delete","Save","Restore","Reset"]  
perm = itertools.combinations(listA,3)  
for i in list(perm):  
    comb = itertools.permutations(i)  
    for j in list(comb):  
        print(j)
```

Python



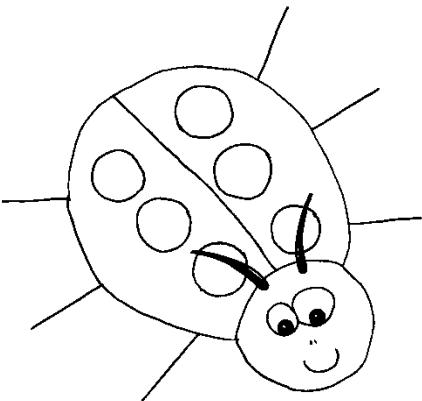
# Combination & Permutation Example

Test	Sequence of operations	Test	Sequence of operations	Test	Sequence of operations	Test	Sequence of operations
TC001	('Create', 'Read', 'Update')	TC026	('Create', 'Reset', 'Read')	TC051	('Update', 'Create', 'Reset')	TC076	('Save', 'Restore', 'Create')
TC002	('Create', 'Update', 'Read')	TC027	('Read', 'Create', 'Reset')	TC052	('Update', 'Reset', 'Create')	TC077	('Restore', 'Create', 'Save')
TC003	('Read', 'Create', 'Update')	TC028	('Read', 'Reset', 'Create')	TC053	('Reset', 'Create', 'Update')	TC078	('Restore', 'Save', 'Create')
TC004	('Read', 'Update', 'Create')	TC029	('Reset', 'Create', 'Read')	TC054	('Reset', 'Update', 'Create')	TC079	('Create', 'Save', 'Reset')
TC005	('Update', 'Create', 'Read')	TC030	('Reset', 'Read', 'Create')	TC055	('Create', 'Delete', 'Save')	TC080	('Create', 'Reset', 'Save')
TC006	('Update', 'Read', 'Create')	TC031	('Create', 'Update', 'Delete')	TC056	('Create', 'Save', 'Delete')	TC081	('Save', 'Create', 'Reset')
TC007	('Create', 'Read', 'Delete')	TC032	('Create', 'Delete', 'Update')	TC057	('Delete', 'Create', 'Save')	TC082	('Save', 'Reset', 'Create')
TC008	('Create', 'Delete', 'Read')	TC033	('Update', 'Create', 'Delete')	TC058	('Delete', 'Save', 'Create')	TC083	('Reset', 'Create', 'Save')
TC009	('Read', 'Create', 'Delete')	TC034	('Update', 'Delete', 'Create')	TC059	('Save', 'Create', 'Delete')	TC084	('Reset', 'Save', 'Create')
TC010	('Read', 'Delete', 'Create')	TC035	('Delete', 'Create', 'Update')	TC060	('Save', 'Delete', 'Create')	TC085	('Create', 'Restore', 'Reset')
TC011	('Delete', 'Create', 'Read')	TC036	('Delete', 'Update', 'Create')	TC061	('Create', 'Delete', 'Restore')	TC086	('Create', 'Reset', 'Restore')
TC012	('Delete', 'Read', 'Create')	TC037	('Create', 'Update', 'Save')	TC062	('Create', 'Restore', 'Delete')	TC087	('Restore', 'Create', 'Reset')
TC013	('Create', 'Read', 'Save')	TC038	('Create', 'Save', 'Update')	TC063	('Delete', 'Create', 'Restore')	TC088	('Restore', 'Reset', 'Create')
TC014	('Create', 'Save', 'Read')	TC039	('Update', 'Create', 'Save')	TC064	('Delete', 'Restore', 'Create')	TC089	('Reset', 'Create', 'Restore')
TC015	('Read', 'Create', 'Save')	TC040	('Update', 'Save', 'Create')	TC065	('Restore', 'Create', 'Delete')	TC090	('Reset', 'Restore', 'Create')
TC016	('Read', 'Save', 'Create')	TC041	('Save', 'Create', 'Update')	TC066	('Restore', 'Delete', 'Create')	TC091	('Read', 'Update', 'Delete')
TC017	('Save', 'Create', 'Read')	TC042	('Save', 'Update', 'Create')	TC067	('Create', 'Delete', 'Reset')	TC092	('Read', 'Delete', 'Update')
TC018	('Save', 'Read', 'Create')	TC043	('Create', 'Update', 'Restore')	TC068	('Create', 'Reset', 'Delete')	TC093	('Update', 'Read', 'Delete')
TC019	('Create', 'Read', 'Restore')	TC044	('Create', 'Restore', 'Update')	TC069	('Delete', 'Create', 'Reset')	TC094	('Update', 'Delete', 'Read')
TC020	('Create', 'Restore', 'Read')	TC045	('Update', 'Create', 'Restore')	TC070	('Delete', 'Reset', 'Create')	TC095	('Delete', 'Read', 'Update')
TC021	('Read', 'Create', 'Restore')	TC046	('Update', 'Restore', 'Create')	TC071	('Reset', 'Create', 'Delete')	TC096	('Delete', 'Update', 'Read')
TC022	('Read', 'Restore', 'Create')	TC047	('Restore', 'Create', 'Update')	TC072	('Reset', 'Delete', 'Create')	TC097	('Read', 'Update', 'Save')
TC023	('Restore', 'Create', 'Read')	TC048	('Restore', 'Update', 'Create')	TC073	('Create', 'Save', 'Restore')	TC098	('Read', 'Save', 'Update')
TC024	('Restore', 'Read', 'Create')	TC049	('Create', 'Update', 'Reset')	TC074	('Create', 'Restore', 'Save')	TC099	('Update', 'Read', 'Save')
TC025	('Create', 'Read', 'Reset')	TC050	('Create', 'Reset', 'Update')	TC075	('Save', 'Create', 'Restore')	TC100	('Update', 'Save', 'Read')



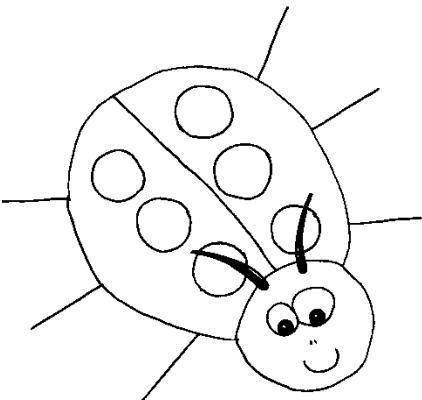
# Combination & Permutation Example

Test	Sequence of operations	Test	Sequence of operations	Test	Sequence of operations	Test	Sequence of operations
TC101	('Save', 'Read', 'Update')	TC126	('Restore', 'Delete', 'Read')	TC151	('Update', 'Delete', 'Save')	TC176	('Update', 'Reset', 'Save')
TC102	('Save', 'Update', 'Read')	TC127	('Read', 'Delete', 'Reset')	TC152	('Update', 'Save', 'Delete')	TC177	('Save', 'Update', 'Reset')
TC103	('Read', 'Update', 'Restore')	TC128	('Read', 'Reset', 'Delete')	TC153	('Delete', 'Update', 'Save')	TC178	('Save', 'Reset', 'Update')
TC104	('Read', 'Restore', 'Update')	TC129	('Delete', 'Read', 'Reset')	TC154	('Delete', 'Save', 'Update')	TC179	('Reset', 'Update', 'Save')
TC105	('Update', 'Read', 'Restore')	TC130	('Delete', 'Reset', 'Read')	TC155	('Save', 'Update', 'Delete')	TC180	('Reset', 'Save', 'Update')
TC106	('Update', 'Restore', 'Read')	TC131	('Reset', 'Read', 'Delete')	TC156	('Save', 'Delete', 'Update')	TC181	('Update', 'Restore', 'Reset')
TC107	('Restore', 'Read', 'Update')	TC132	('Reset', 'Delete', 'Read')	TC157	('Update', 'Delete', 'Restore')	TC182	('Update', 'Reset', 'Restore')
TC108	('Restore', 'Update', 'Read')	TC133	('Read', 'Save', 'Restore')	TC158	('Update', 'Restore', 'Delete')	TC183	('Restore', 'Update', 'Reset')
TC109	('Read', 'Update', 'Reset')	TC134	('Read', 'Restore', 'Save')	TC159	('Delete', 'Update', 'Restore')	TC184	('Restore', 'Reset', 'Update')
TC110	('Read', 'Reset', 'Update')	TC135	('Save', 'Read', 'Restore')	TC160	('Delete', 'Restore', 'Update')	TC185	('Reset', 'Update', 'Restore')
TC111	('Update', 'Read', 'Reset')	TC136	('Save', 'Restore', 'Read')	TC161	('Restore', 'Update', 'Delete')	TC186	('Reset', 'Restore', 'Update')
TC112	('Update', 'Reset', 'Read')	TC137	('Restore', 'Read', 'Save')	TC162	('Restore', 'Delete', 'Update')	TC187	('Delete', 'Save', 'Restore')
TC113	('Reset', 'Read', 'Update')	TC138	('Restore', 'Save', 'Read')	TC163	('Update', 'Delete', 'Reset')	TC188	('Delete', 'Restore', 'Save')
TC114	('Reset', 'Update', 'Read')	TC139	('Read', 'Save', 'Reset')	TC164	('Update', 'Reset', 'Delete')	TC189	('Save', 'Delete', 'Restore')
TC115	('Read', 'Delete', 'Save')	TC140	('Read', 'Reset', 'Save')	TC165	('Delete', 'Update', 'Reset')	TC190	('Save', 'Restore', 'Delete')
TC116	('Read', 'Save', 'Delete')	TC141	('Save', 'Read', 'Reset')	TC166	('Delete', 'Reset', 'Update')	TC191	('Restore', 'Delete', 'Save')
TC117	('Delete', 'Read', 'Save')	TC142	('Save', 'Reset', 'Read')	TC167	('Reset', 'Update', 'Delete')	TC192	('Restore', 'Save', 'Delete')
TC118	('Delete', 'Save', 'Read')	TC143	('Reset', 'Read', 'Save')	TC168	('Reset', 'Delete', 'Update')	TC193	('Delete', 'Save', 'Reset')
TC119	('Save', 'Read', 'Delete')	TC144	('Reset', 'Save', 'Read')	TC169	('Update', 'Save', 'Restore')	TC194	('Delete', 'Reset', 'Save')
TC120	('Save', 'Delete', 'Read')	TC145	('Read', 'Restore', 'Reset')	TC170	('Update', 'Restore', 'Save')	TC195	('Save', 'Delete', 'Reset')
TC121	('Read', 'Delete', 'Restore')	TC146	('Read', 'Reset', 'Restore')	TC171	('Save', 'Update', 'Restore')	TC196	('Save', 'Reset', 'Delete')
TC122	('Read', 'Restore', 'Delete')	TC147	('Restore', 'Read', 'Reset')	TC172	('Save', 'Restore', 'Update')	TC197	('Reset', 'Delete', 'Save')
TC123	('Delete', 'Read', 'Restore')	TC148	('Restore', 'Reset', 'Read')	TC173	('Restore', 'Update', 'Save')	TC198	('Reset', 'Save', 'Delete')
TC124	('Delete', 'Restore', 'Read')	TC149	('Reset', 'Read', 'Restore')	TC174	('Restore', 'Save', 'Update')	TC199	('Delete', 'Restore', 'Reset')
TC125	('Restore', 'Read', 'Delete')	TC150	('Reset', 'Restore', 'Read')	TC175	('Update', 'Save', 'Reset')	TC200	('Delete', 'Reset', 'Restore')



# Combination & Permutation Example

Test	Sequence of operations
TC201	('Restore', 'Delete', 'Reset')
TC202	('Restore', 'Reset', 'Delete')
TC203	('Reset', 'Delete', 'Restore')
TC204	('Reset', 'Restore', 'Delete')
TC205	('Save', 'Restore', 'Reset')
TC206	('Save', 'Reset', 'Restore')
TC207	('Restore', 'Save', 'Reset')
TC208	('Restore', 'Reset', 'Save')
TC209	('Reset', 'Save', 'Restore')
TC210	('Reset', 'Restore', 'Save')



# Multi-Variable Variation Example

Transaction

Create

Read

Update

Delete

3 Variables

Location

International

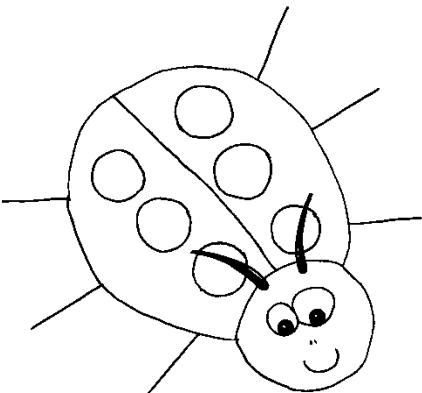
Domestic

Customer

New

Existing

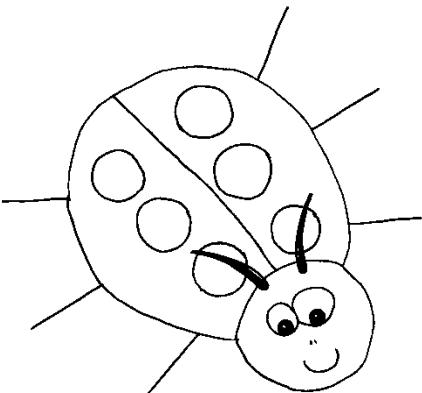
Try all possible variations of each



# Multi-Variable Variation Example

```
import itertools  
listA = ["Create","Read","Update","Delete"]  
listB = ["International","Domestic"]  
listC = ["New","Old"]  
perm = itertools.product(listA,listB,listC)  
for i in list(perm):  
    print(i)
```

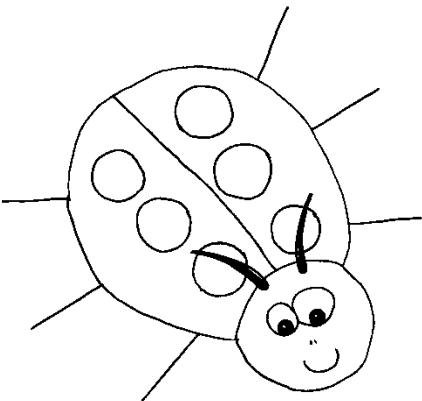
Python



# Multi-Variable Variation Example

```
('Create', 'International', 'New')
('Create', 'International', 'Old')
('Create', 'Domestic', 'New')
('Create', 'Domestic', 'Old')
('Read', 'International', 'New')
('Read', 'International', 'Old')
('Read', 'Domestic', 'New')
('Read', 'Domestic', 'Old')
('Update', 'International', 'New')
('Update', 'International', 'Old')
('Update', 'Domestic', 'New')
('Update', 'Domestic', 'Old')
('Delete', 'International', 'New')
('Delete', 'International', 'Old')
('Delete', 'Domestic', 'New')
('Delete', 'Domestic', 'Old')
```

16 Variations



# Combinations with Repetitions Example

## Transaction

Create

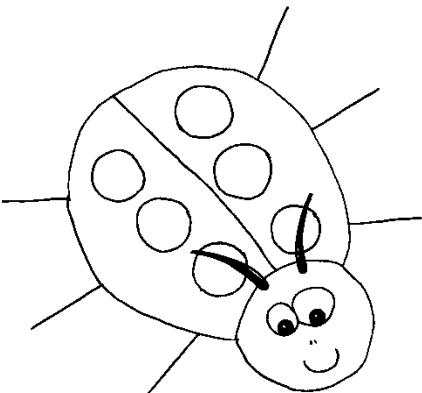
Read

Update

Delete

4  
Transactions

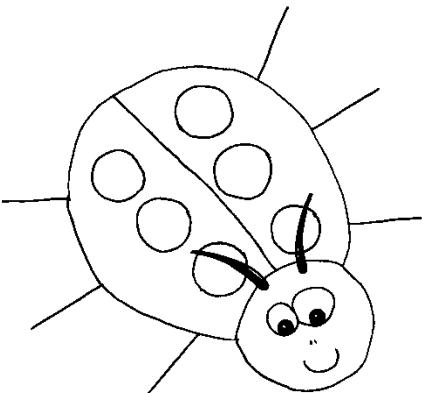
Try all combinations  
of 3 with repetitions



# Combinations with Repetitions Example

```
import itertools  
listA = ["Create","Read","Update","Delete"]  
perm = itertools.combinations_with_replacement(listA,3)  
for i in list(perm):  
    print(i)
```

Python

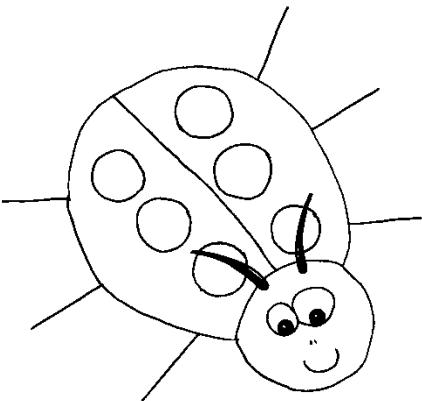


# Combinations with Repetitions Example

('Create', 'Create', 'Create')  
('Create', 'Create', 'Read')  
('Create', 'Create', 'Update')  
('Create', 'Create', 'Delete')  
('Create', 'Read', 'Read')  
('Create', 'Read', 'Update')  
('Create', 'Read', 'Delete')  
('Create', 'Update', 'Update')  
('Create', 'Update', 'Delete')  
('Create', 'Delete', 'Delete')  
('Read', 'Read', 'Read')

('Read', 'Read', 'Update')  
('Read', 'Read', 'Delete')  
('Read', 'Update', 'Update')  
('Read', 'Update', 'Delete')  
('Read', 'Delete', 'Delete')  
('Update', 'Update', 'Update')  
('Update', 'Update', 'Delete')  
('Update', 'Delete', 'Delete')  
('Delete', 'Delete', 'Delete')

20  
Combinations



# Installer Test Variations Example

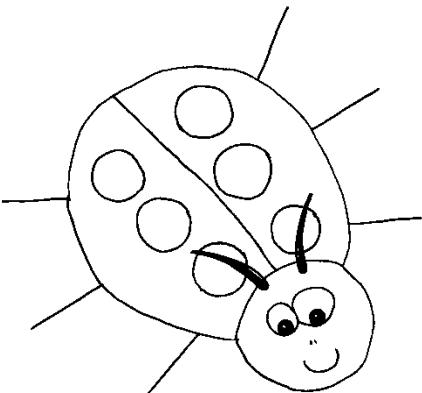
## Options

A .. J

Choose 2

10  
Transactions

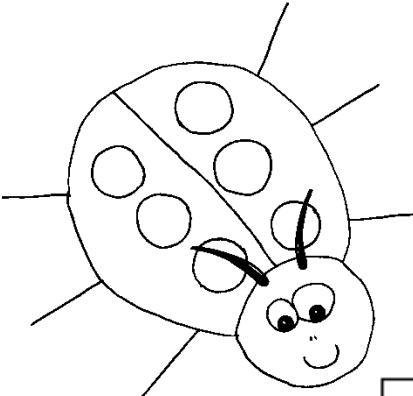
Vary two options in  
each trial



# Installer Test Variations Example

```
import itertools  
listA = ["A","B","C","D","E","F","G","H","I","J"]  
perm = itertools.combinations(listA,2)  
for i in list(perm):  
    print(i)
```

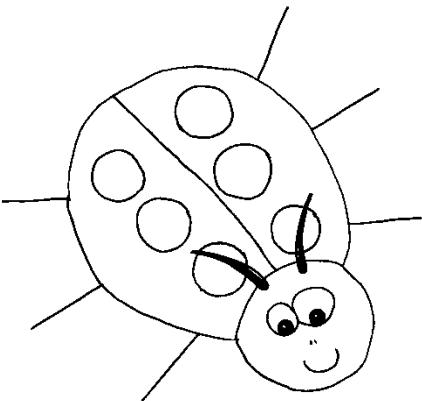
Python



# Installer Test Variations Example

45 Trials

TC001	('A', 'B')	TC016	('B', 'I')	TC031	('E', 'F')
TC002	('A', 'C')	TC017	('B', 'J')	TC032	('E', 'G')
TC003	('A', 'D')	TC018	('C', 'D')	TC033	('E', 'H')
TC004	('A', 'E')	TC019	('C', 'E')	TC034	('E', 'I')
TC005	('A', 'F')	TC020	('C', 'F')	TC035	('E', 'J')
TC006	('A', 'G')	TC021	('C', 'G')	TC036	('F', 'G')
TC007	('A', 'H')	TC022	('C', 'H')	TC037	('F', 'H')
TC008	('A', 'I')	TC023	('C', 'I')	TC038	('F', 'I')
TC009	('A', 'J')	TC024	('C', 'J')	TC039	('F', 'J')
TC010	('B', 'C')	TC025	('D', 'E')	TC040	('G', 'H')
TC011	('B', 'D')	TC026	('D', 'F')	TC041	('G', 'I')
TC012	('B', 'E')	TC027	('D', 'G')	TC042	('G', 'J')
TC013	('B', 'F')	TC028	('D', 'H')	TC043	('H', 'I')
TC014	('B', 'G')	TC029	('D', 'I')	TC044	('H', 'J')
TC015	('B', 'H')	TC030	('D', 'J')	TC045	('I', 'J')



# Installer Random Variations Example

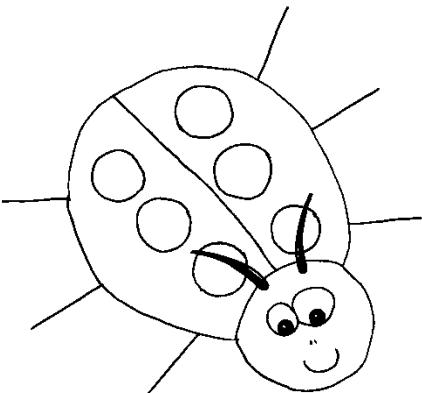
## Options

A .. J

Choose 2

10  
Transactions

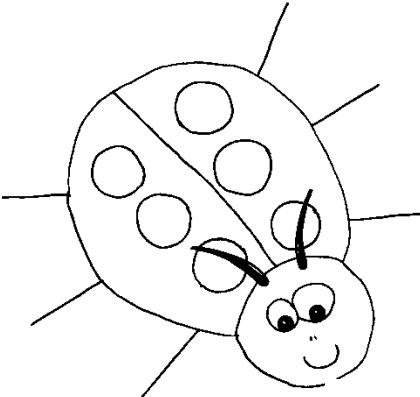
Randomly vary two  
options in each trial



# Installer Random Variations Example

```
import itertools  
import random  
  
listA = ["A","B","C","D","E","F","G","H","I","J"]  
perm = list(itertools.combinations(listA,2))  
random.shuffle(perm)  
  
for i in perm:  
    print(i)
```

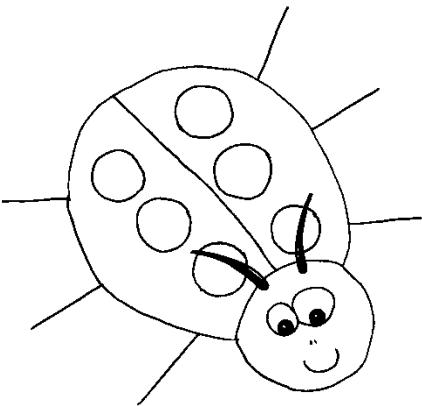
Python



# Installer Random Variations Example

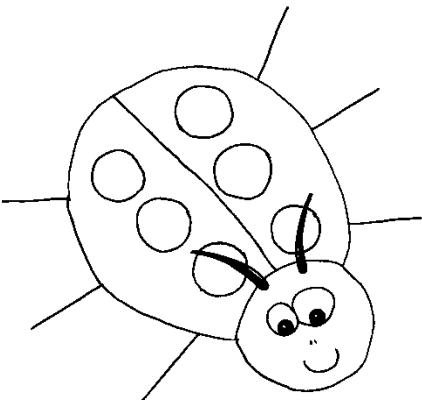
45 Random Trials

TC001	('G', 'I')	TC016	('D', 'E')	TC031	('C', 'G')
TC002	('G', 'H')	TC017	('E', 'I')	TC032	('H', 'I')
TC003	('A', 'G')	TC018	('E', 'J')	TC033	('B', 'C')
TC004	('B', 'E')	TC019	('A', 'C')	TC034	('C', 'E')
TC005	('B', 'G')	TC020	('B', 'H')	TC035	('A', 'J')
TC006	('E', 'H')	TC021	('B', 'J')	TC036	('C', 'J')
TC007	('A', 'H')	TC022	('C', 'I')	TC037	('E', 'F')
TC008	('C', 'D')	TC023	('F', 'G')	TC038	('F', 'J')
TC009	('B', 'I')	TC024	('E', 'G')	TC039	('A', 'I')
TC010	('A', 'D')	TC025	('A', 'E')	TC040	('D', 'I')
TC011	('H', 'J')	TC026	('D', 'J')	TC041	('B', 'D')
TC012	('F', 'H')	TC027	('C', 'F')	TC042	('B', 'F')
TC013	('D', 'G')	TC028	('F', 'I')	TC043	('I', 'J')
TC014	('D', 'F')	TC029	('D', 'H')	TC044	('A', 'B')
TC015	('A', 'F')	TC030	('C', 'H')	TC045	('G', 'J')



# Test Design

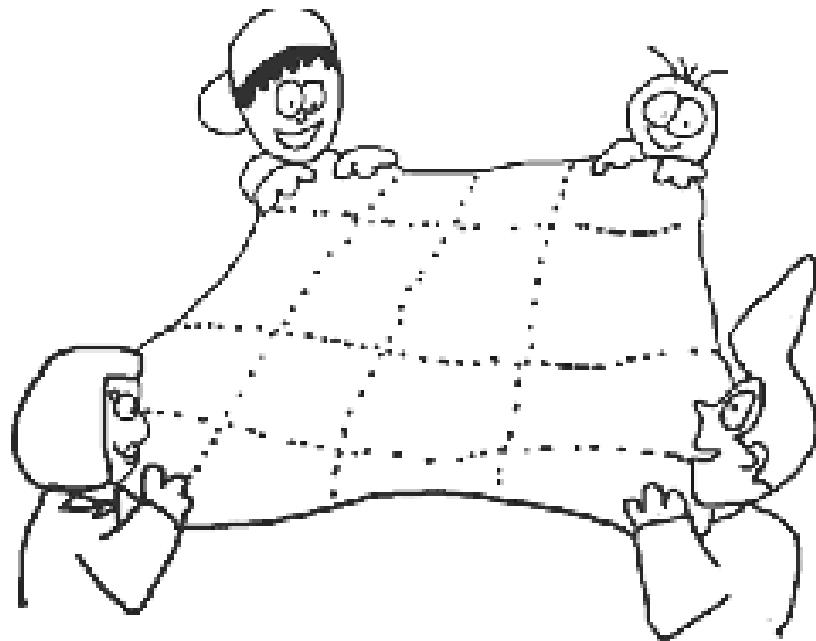
*Code Coverage*

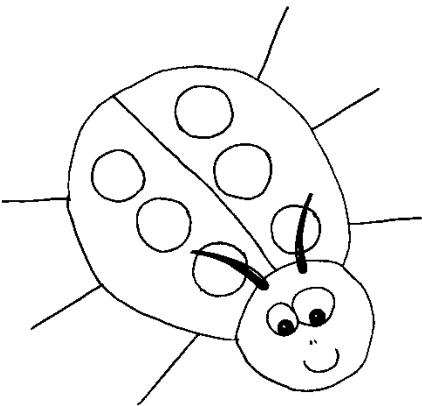


# Code Coverage

## Code Coverage

- Statement (Line)
- Branch
- Condition (Decision)
- Function
- Path
  - All Nodes
  - All Edges
  - All Paths
  - All Basis Paths
  - Data Flow

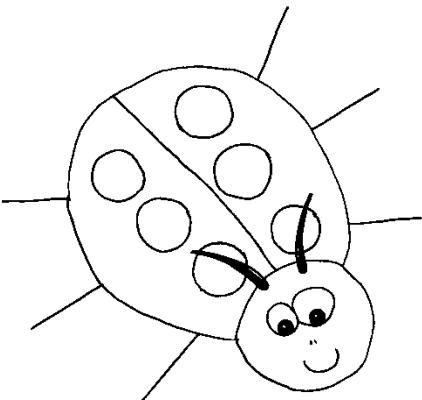




# Code Coverage

## Code Coverage Risks

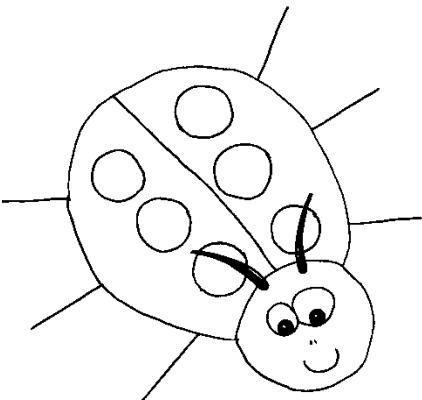
- Focus on solution
- Exercise code that was written
- Do not exercise code that was omitted



# Code Coverage

## Code Coverage Risks

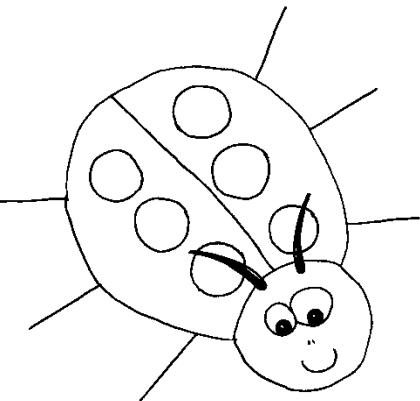
- False sense of confidence
- False sense of completeness
- False sense of Quality



# Code Coverage

## Code Coverage Popularity

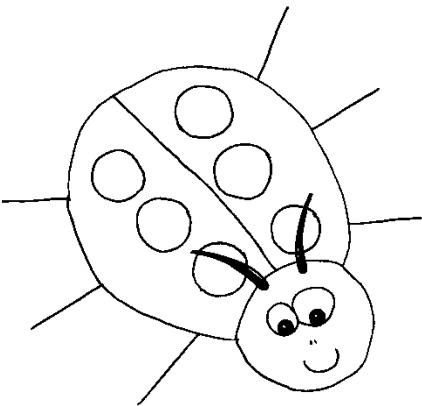
- Regression Test Dashboards
- Continuous Integration
- DevOps
- Tool Support



# Code Coverage

## Statement (Line)

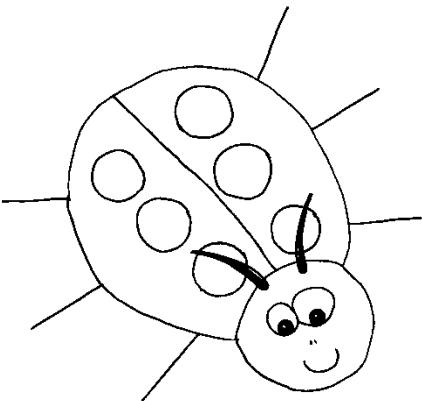
- 100% Statement Coverage implies each statement has been executed at least once



# Code Coverage

## Branch

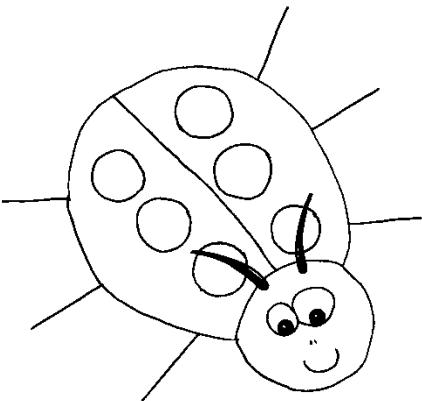
- 100% Branch Coverage implies each possible outcome of each decision has been executed at least once



# Code Coverage

## Condition (Decision)

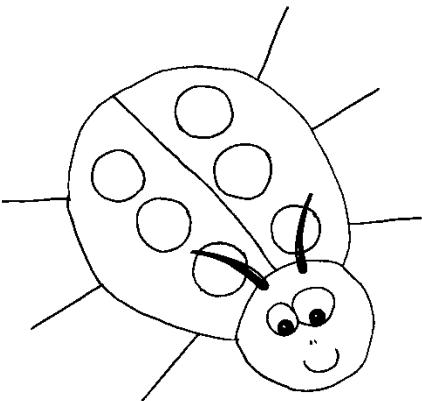
- 100% Condition Coverage implies each possible way a decision can be TRUE and each possible way a decision can be FALSE has been exercised at least once



# Code Coverage

## Function

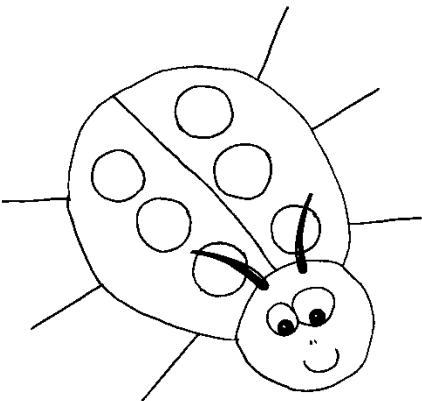
- 100% Function Coverage implies each possible “Function”, “Method”, “Process”, “Procedure” or other executable element of the code has been exercised at least once



# Code Coverage

## Path

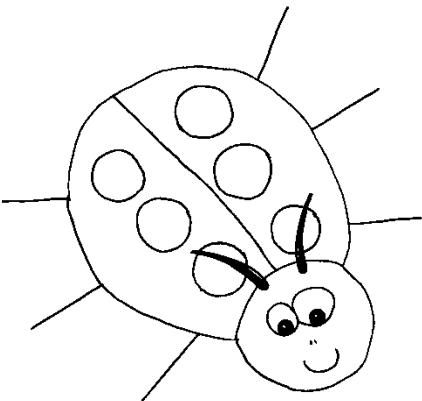
- Path coverage models code using a directed graphs in which a node represents one or more statements and an edge represents the flow of code



# Code Coverage

## Path “All Nodes”

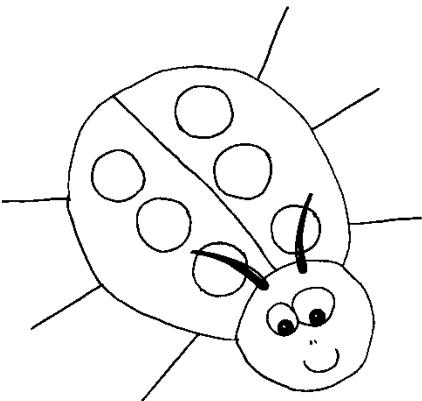
- 100% Node Coverage implies each possible node in the directed graph representing the code has been exercised at least once



# Code Coverage

## Path “All Edges”

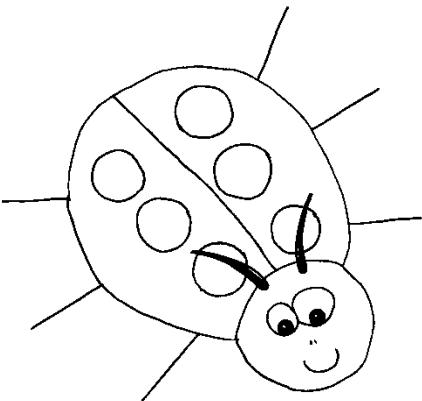
- 100% Edge Coverage implies each possible edge in the directed graph representing the code has been exercised at least once



# Code Coverage

## Path “All Paths”

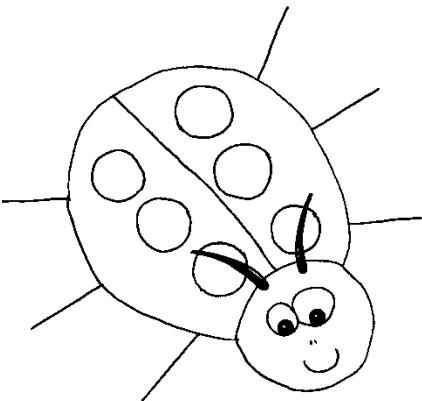
- 100% Path Coverage implies each possible path through the directed graph representing the code has been exercised at least once



# Code Coverage

## Path “All Basis Paths”

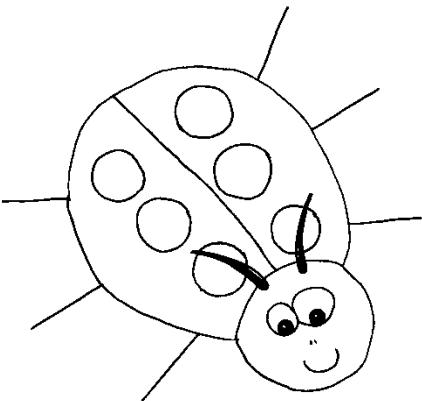
- 100% Basis Path Coverage implies each possible basis path through the directed graph representing the code has been exercised at least once



# Code Coverage

## Path “Data Flow”

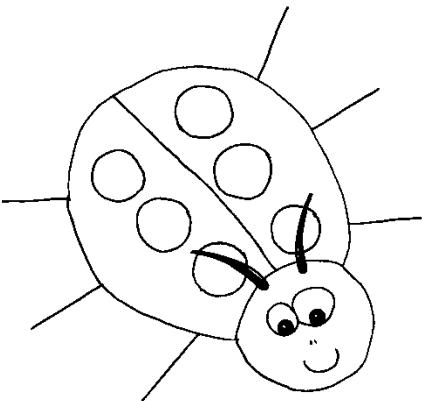
- 100% Data Flow Coverage implies that each path, which uses data, through the directed graph representing the code has been exercised at least once



# Code Coverage

## Trivia

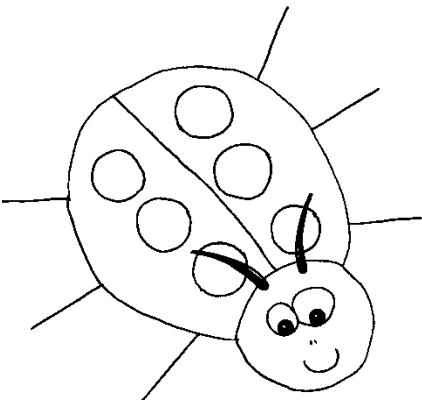
- 100% Statement Coverage does not guarantee 100% Branch Coverage



# Code Coverage

## Trivia

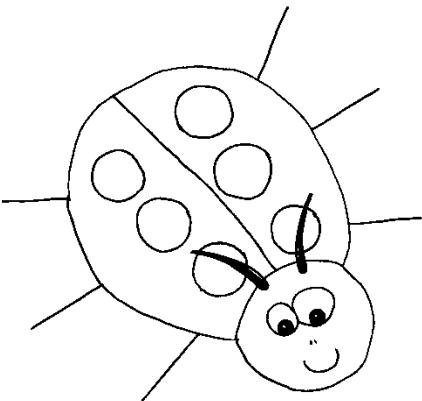
- 100% Branch Coverage does not guarantee 100% “Path” Coverage



# Code Coverage

## Trivia

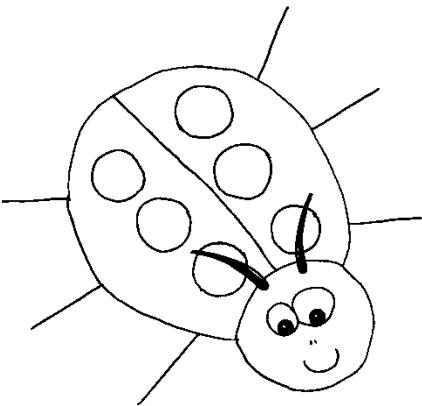
- 100% “Basis Path”  
Coverage guarantees  
100% “Statement”  
Coverage



# Code Coverage

## Trivia

- 100% “Basis Path” Coverage guarantees 100% “Branch” Coverage

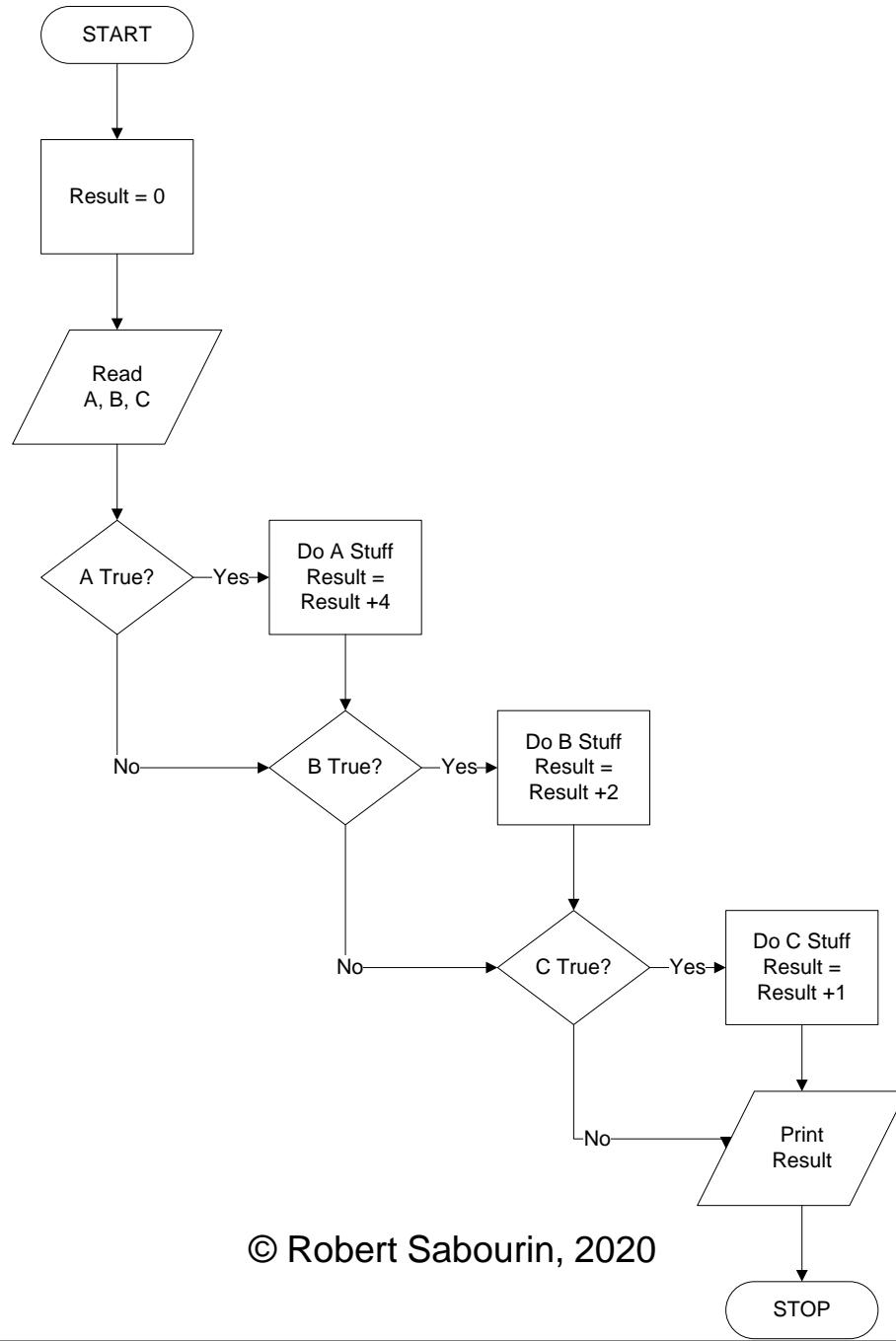
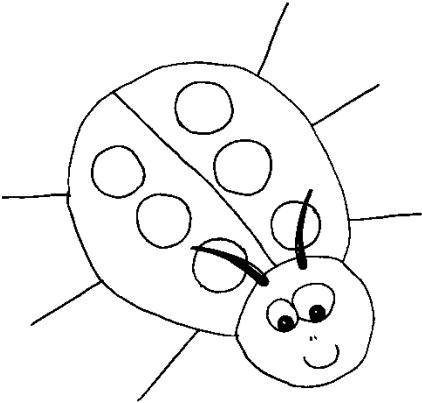


# Code Coverage

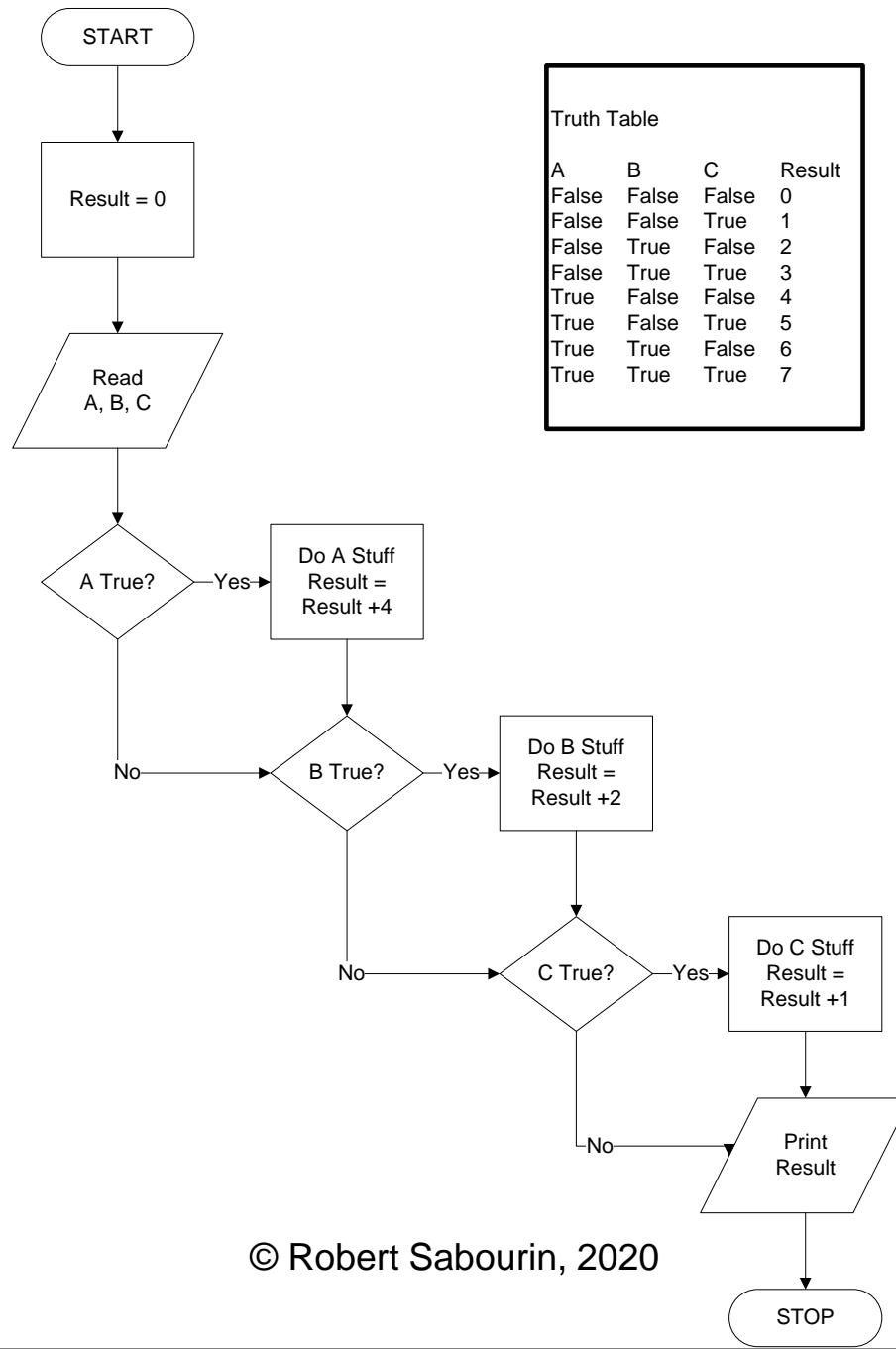
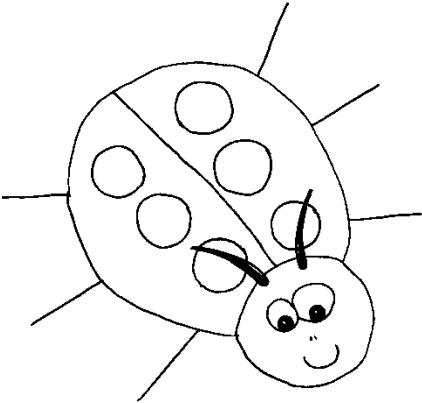
## Trivia

- 100% “Path” Coverage  
may be impossible for  
example looping in code

# Go With the Flow!

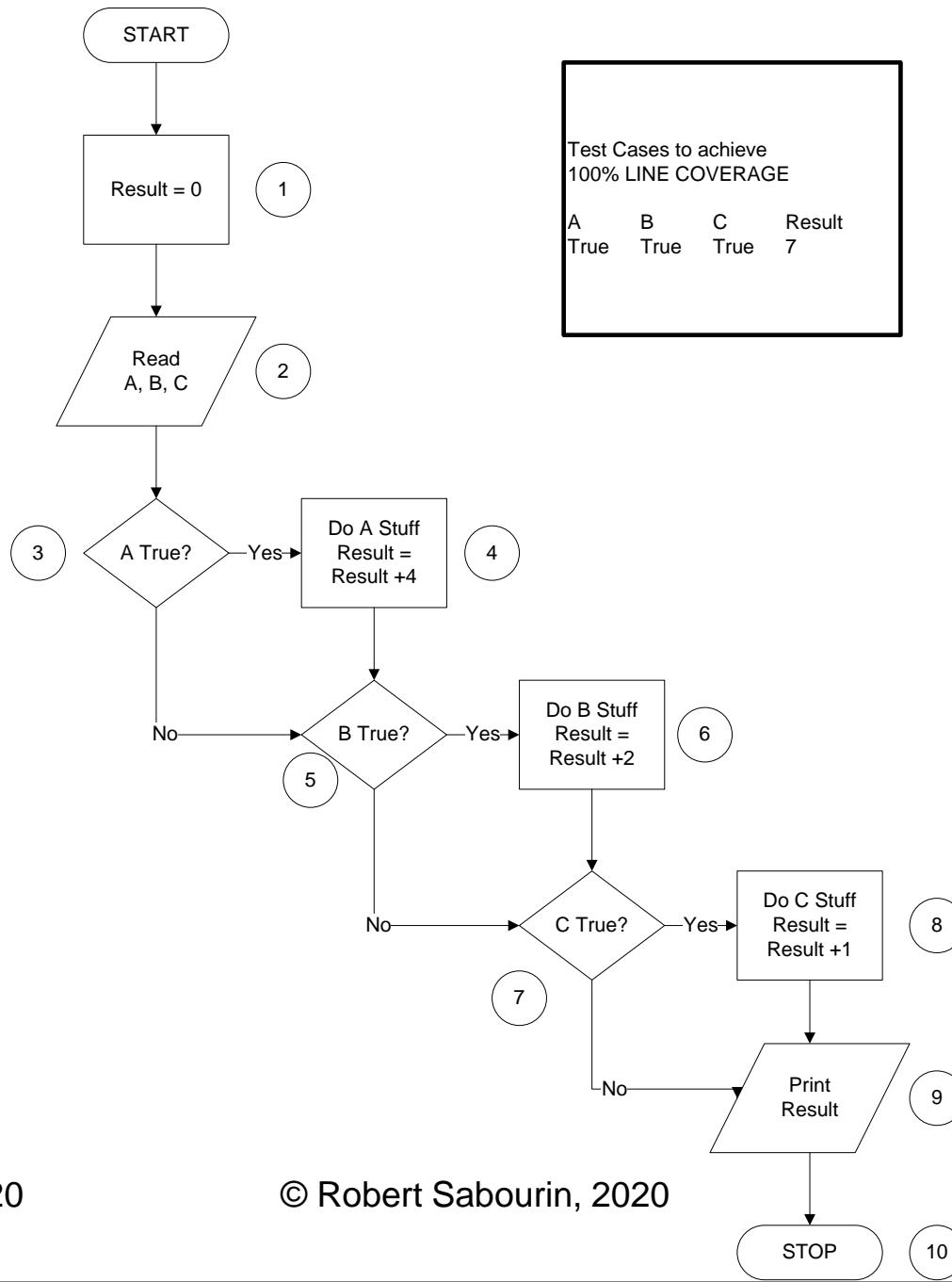
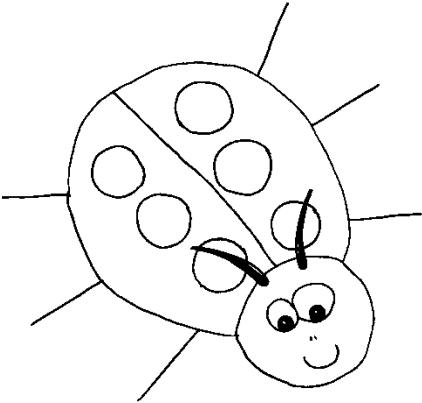


# Go With the Flow!



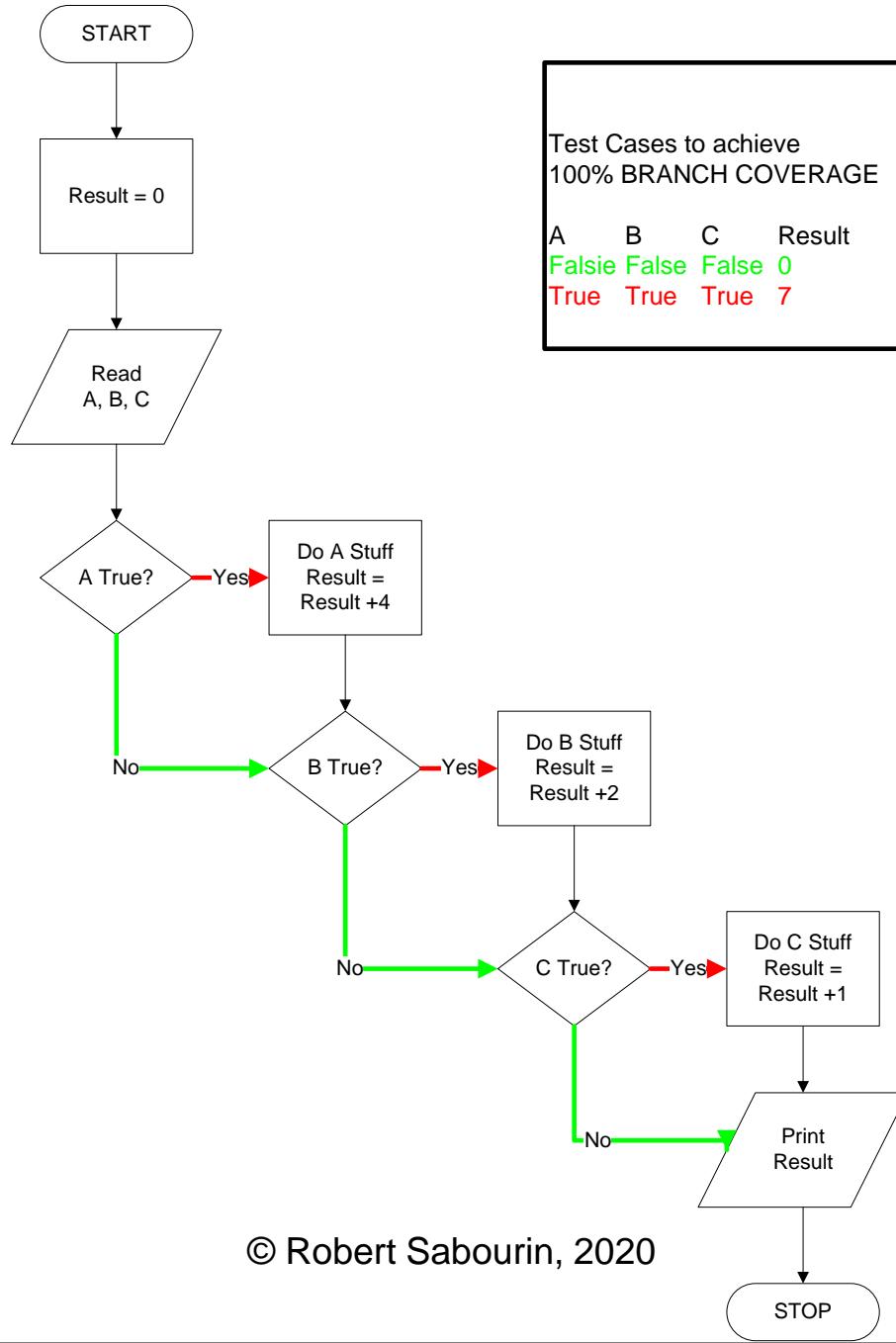
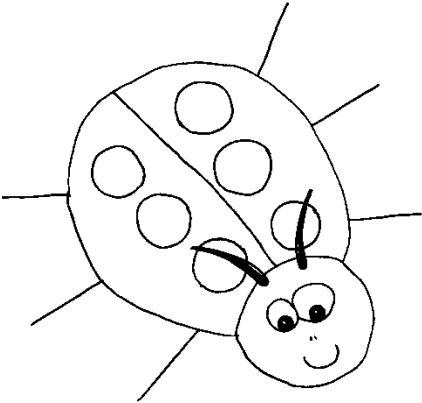
# Testing Coverage

## Go With the Flow!



# Testing Coverage

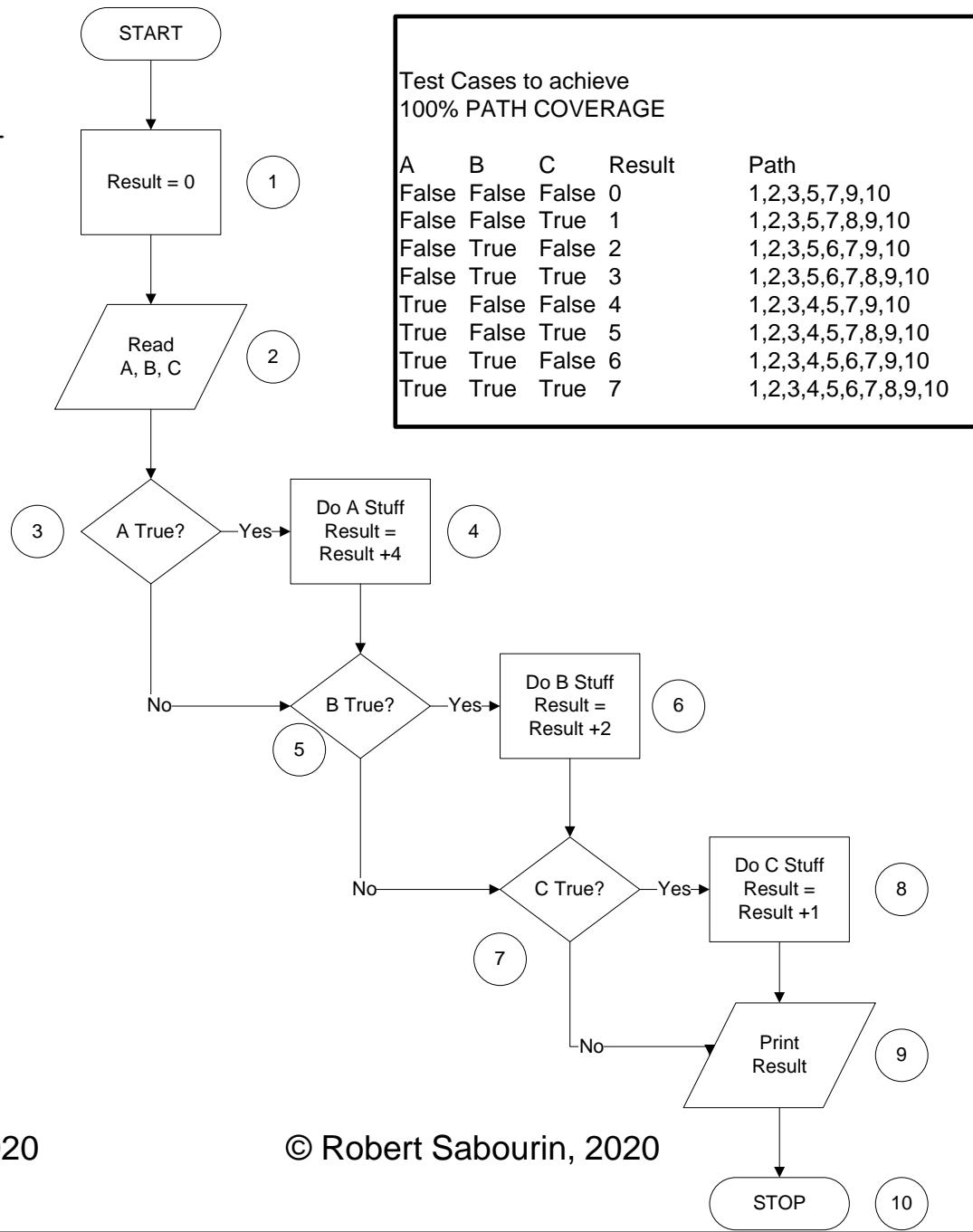
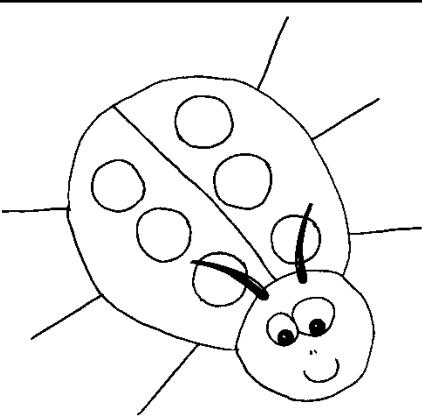
## Go With the Flow!

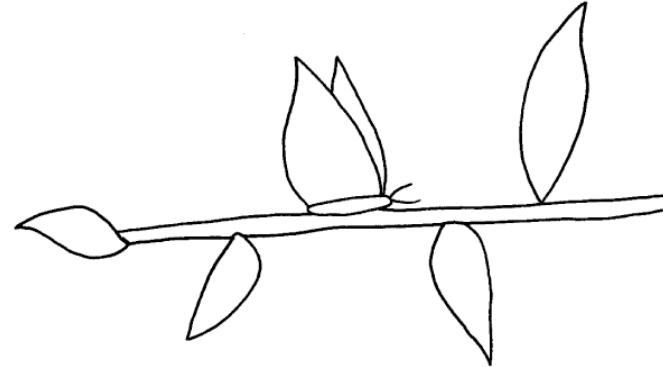
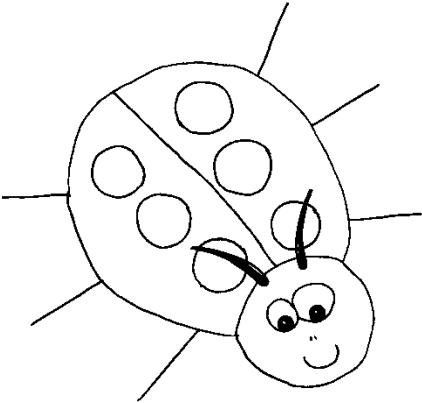


Test Cases to achieve  
100% BRANCH COVERAGE

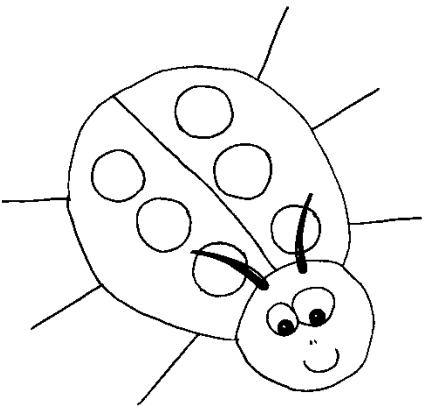
A	B	C	Result
False	False	False	0
True	True	True	7

## Go With the Flow!



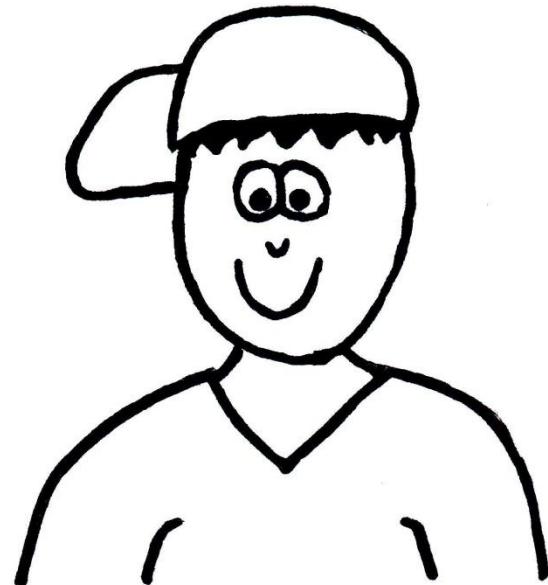
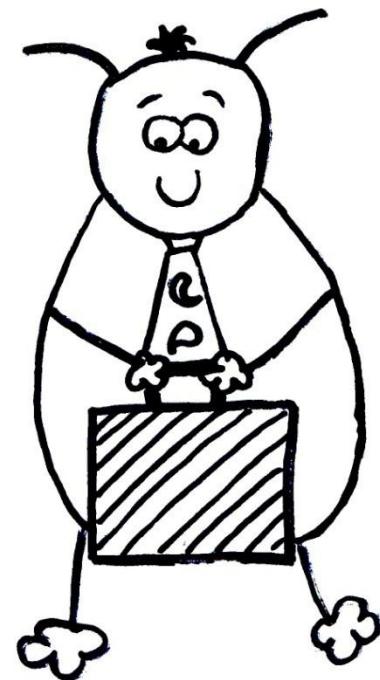


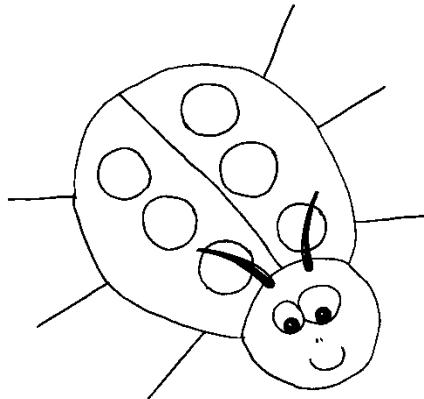
What about  
bugs of  
omission?



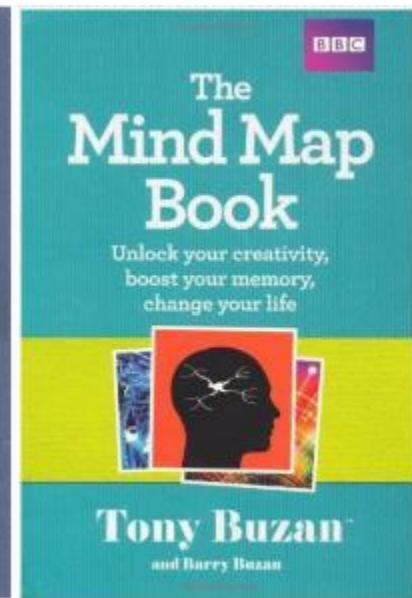
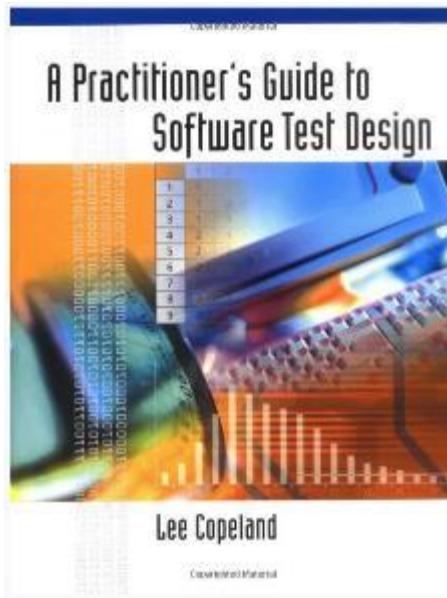
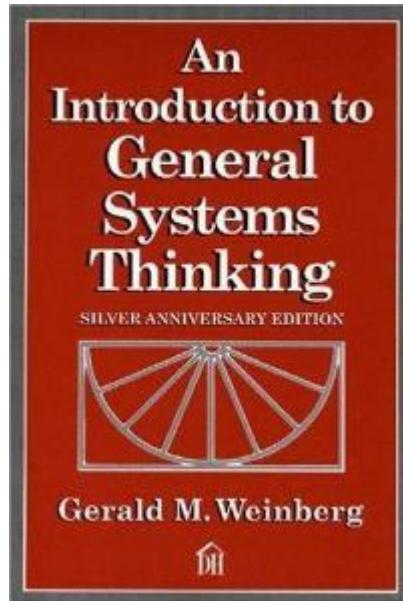
# Thank You

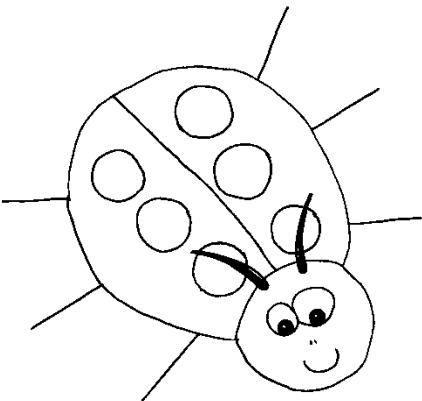
- Questions?



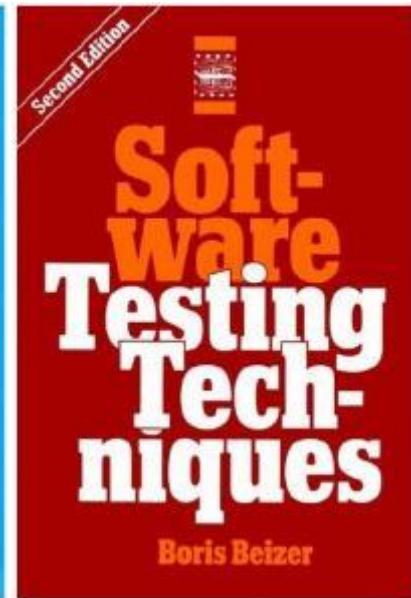
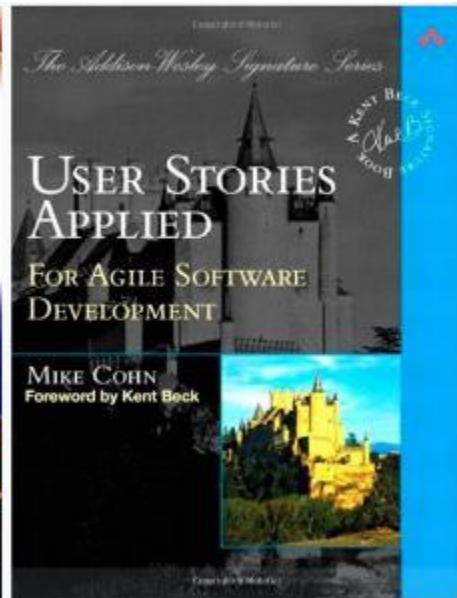
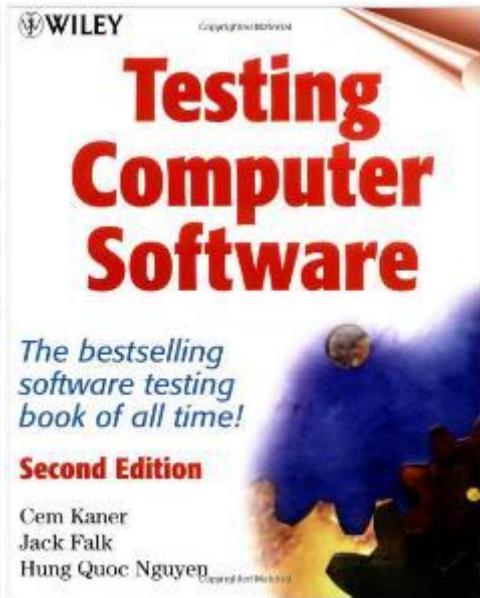
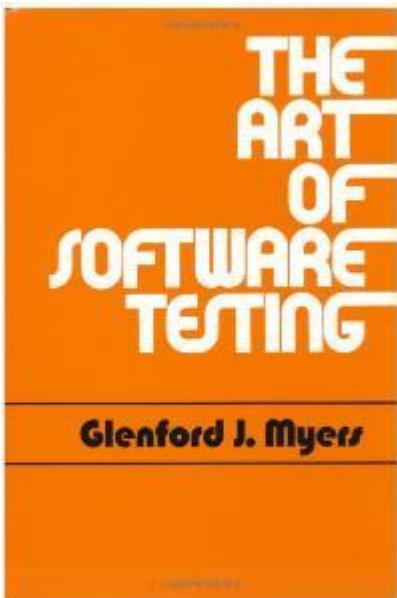


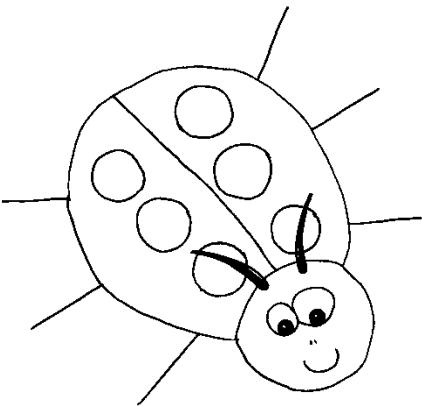
# Some References





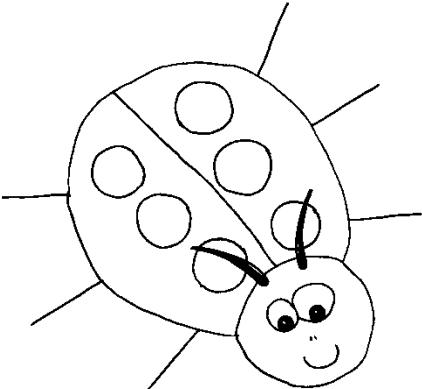
# Some References





# Some References





# *Some Tools*

**Fitnessse**

<http://fitnessse.org/>

---

**Cucumber**

<http://cukes.info/>

---

**Freemind**

[http://freemind.sourceforge.net/wiki/index.php/Main\\_Page](http://freemind.sourceforge.net/wiki/index.php/Main_Page)

---

**PICT**

<http://download.microsoft.com/download/f/5/5/f55484df-8494-48fa-8dbd-8c6f76cc014b/pict33.msi>

---

**AllPairs**

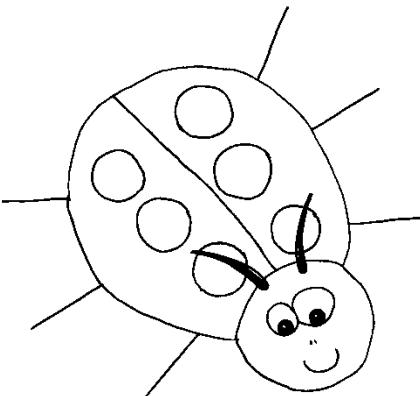
<http://www.satisfice.com/testmethod.shtml>

---

**e2gRuleWriter**

<http://expertise2go.com/e2g3g/e2g3gdoc/e2gRuleWriterRef.htm#SOFTWARE>

---



# Some Web References

Gmail Examples

[http://www.amibugshare.com/examples/Gmail\\_test.zip](http://www.amibugshare.com/examples/Gmail_test.zip)

PICT Online

<http://www.amibugshare.com/pict/>

Mind Mapping

[http://www.amibugshare.com/examples/Mind\\_Mapping.zip](http://www.amibugshare.com/examples/Mind_Mapping.zip)

Articles

<http://www.amibugshare.com/articles/>

Examples

<http://www.amibugshare.com/examples/>

Case Studies

[http://www.amibugshare.com/case\\_studies/](http://www.amibugshare.com/case_studies/)