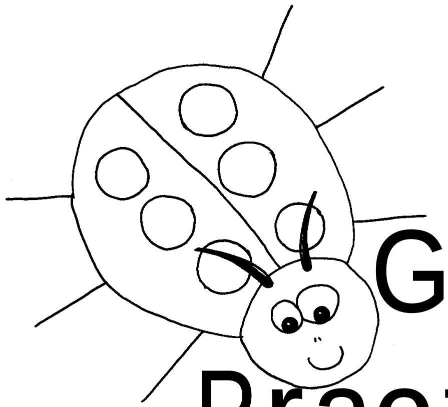


Getting Things Done: Practical Web/e-Commerce Application Stress Testing

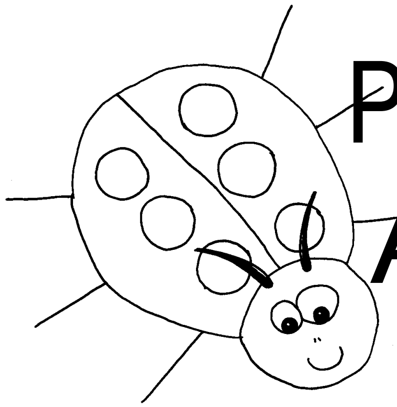
Robert Sabourin

President

AmiBug.Com, Inc.

Montreal, Canada

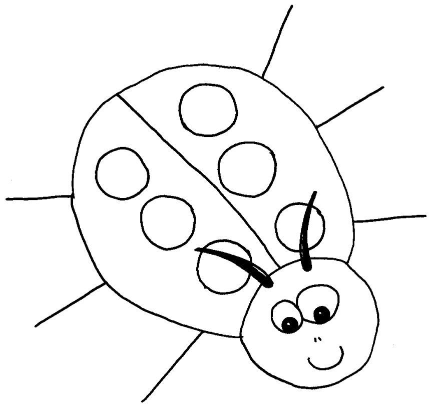
rsabourin@amibug.com



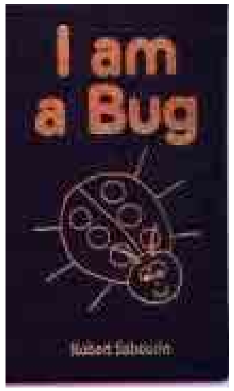
Practical Web/e-Commerce Application Stress Testing

Overview:

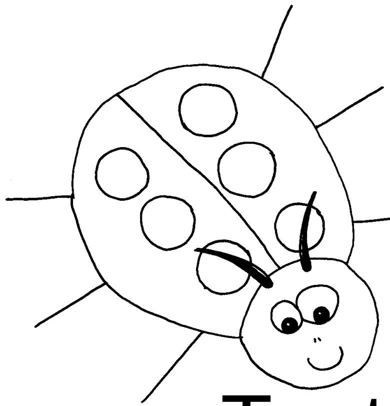
- Definitions
- Stress Testing & Development Process
- Stress Testing & Requirements
- Stress Testing & Analysis - Design
- Stress Testing & Development
- Stress Testing & Testing Process
- Stress Testing Tools & Services



Practical Web/e-Commerce Application Stress Testing

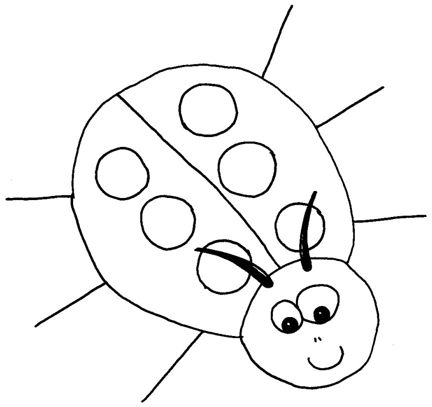


- Robert Sabourin , *Software Evangelist*
- President
- AmiBug.Com Inc.
- Montreal, Quebec, Canada
- rsabourin@amibug.com

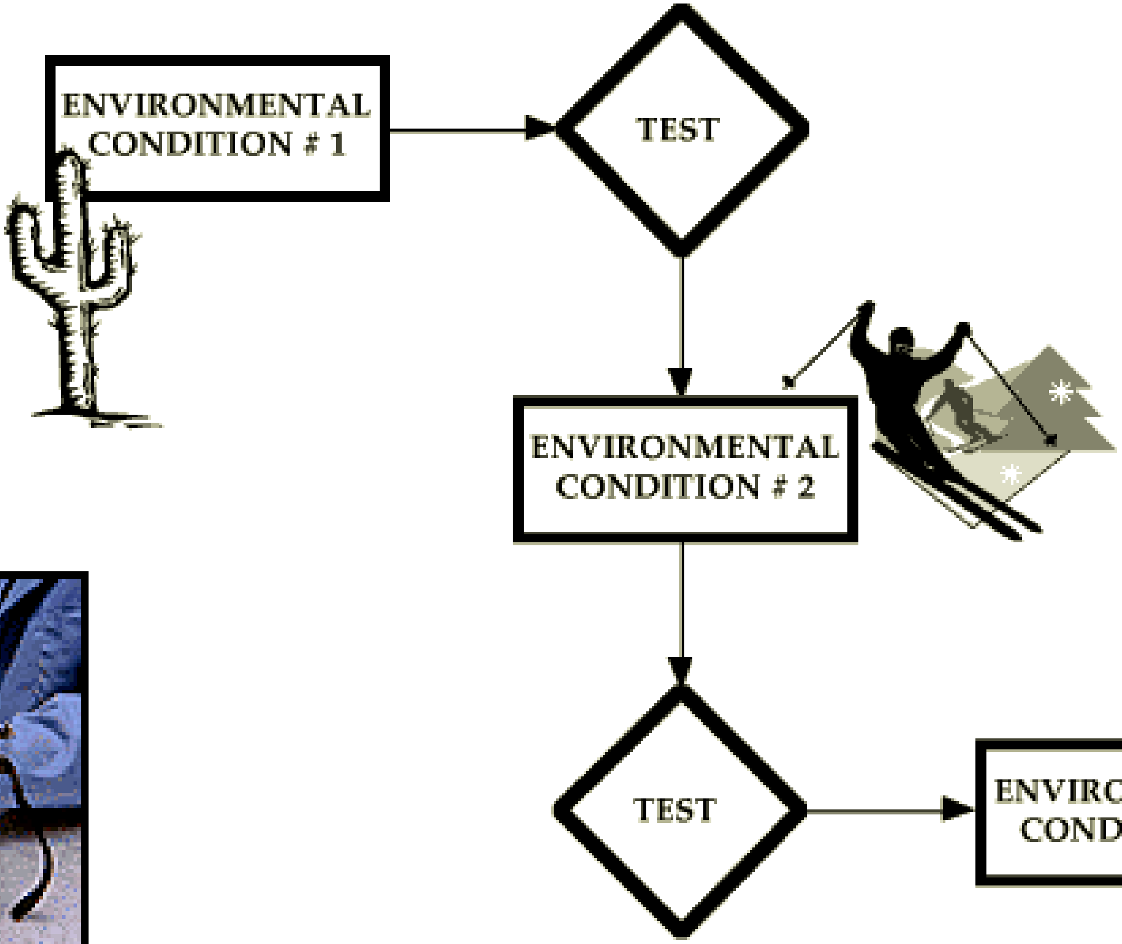


What is Stress Testing?

- Testing operational characteristics of an application within a harshly constrained <u>*environment*</u>
  - *Limit processor speed*
  - *Low memory, disk space*
  - *Diminished access to shared resources*
  - *Physical Environment, Static, Temperature, Humidity*



Stress Testing Embedded Software


ENVIRONMENTAL CONDITION # 1
TEST
ENVIRONMENTAL CONDITION # 2
TEST
ENVIRONMENTAL CONDITION # 3

Rational Unified Process (RUP)

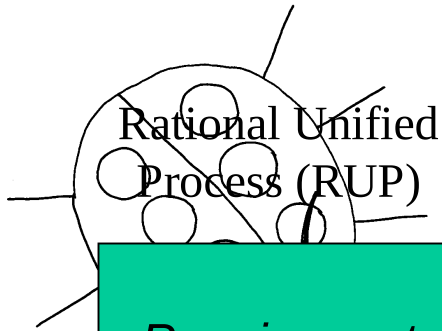
Phase

Inception	*Elaboration*	*Construction*	*Transition*

Stress Testing can take place as part of each phase of development .

Core Workflow

- *Requirements*
- *Analysis*
- *Design*
- *Development*
- *Testing*
- *Maintenance*



Rational Unified Process (RUP)

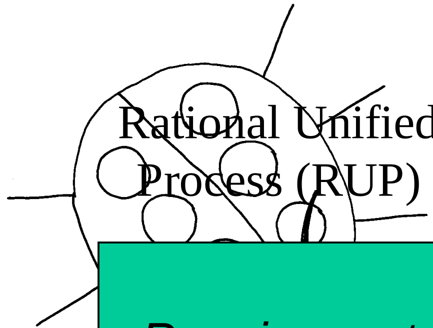
Phase

Inception	*Elaboration*	*Construction*	*Transition*

Core Workflow

- *Requirements*
- *Analysis*
- *Design*
- *Development*
- *Testing*
- *Maintenance*

Stress Testing can take place as part of each core workflow involved in development organization.

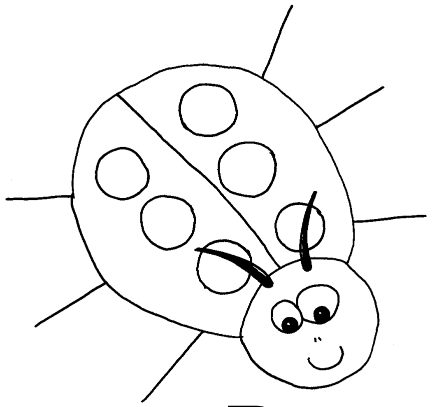


Rational Unified Process (RUP)

Phase

Core Workflow

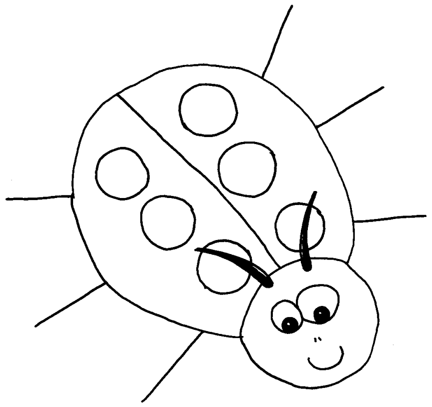
	Inception	Elaboration	Construction	Transition
Requirements	Requirements: Performance Load Scalability Cost, System Constraints Formal inspections		Requirements: Adapt as required New technologies New business needs New budget Changing System Constraints	
Analysis	Architecture: Computer Simulation Prototyping Design alternatives Fall backs		Architecture: Root cause failure analysis Re-factor architecture Adapt to new technologies Reconcile simulation vs.. actual	
Design	Design: Reliability Weakest link Testability hooks Robust middleware Cost tradeoffs		Design: Identify weakest link Complexity Reassess reliable new technology Adapt to changes Hooks to facilitate stress testing	
Development	Development: Develop unit test harness Develop test case repository Determine physical components to stress and define strategy		Development: Use test harness to perform stress testing on developed code on a task assignment basis As required fix bugs found in test	
Testing	Test Team: Work with development to develop test case repository Support developing test harness Set up test lab and build strategy Plan for all measuring methods		Integration: Stress test as new modules come on line System: Stress test early in lab environment Live: Work with site monitoring team	
Maintenance	Preparing: Plan upgrade strategy Study usage patterns Security breaches Customer service Identify site monitor partners		Monitoring: Site performance under load Server memory usage Server processor usage Database usage System resources	



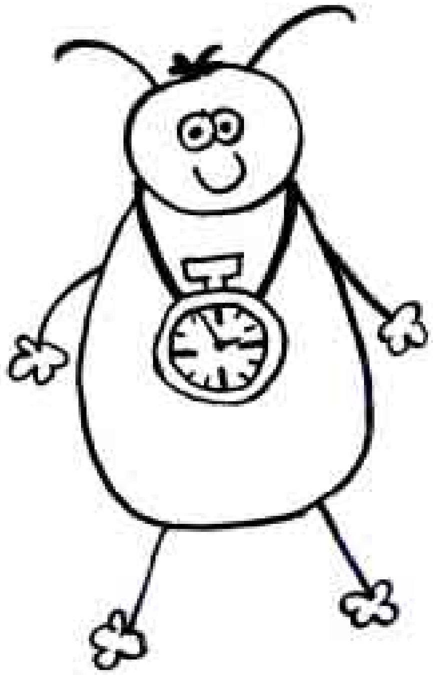
Requirements

- Response time, end user experience
  - Slow vs High Speed connections
- Number of concurrent users
  - Normal vs. Peak
  - Doing what?
- Performance Degradation
- Reliability
  - MTTR, MTTF

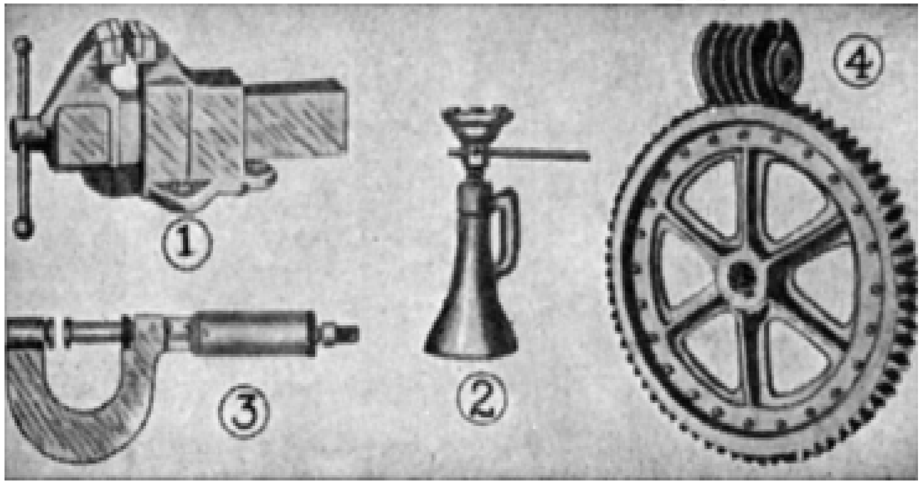
Attributes	*Stress Impact*
Usability	Performance degradation Ensure delays never exceed about 8-10 seconds independent of load
Scalability	Additional resources should increase capacity Vertical or horizontal scaling. Bandwidth, Memory, Processing Power, CPUs, Servers
Serviceability	Success will lead to increased load needing upgrades If underlying components are upgraded does the system still have the same reliability?
Reliability	What is the time to failure, can we predict it? MTTF How does the system act when a process or thread fails? Do we recover?
Maintainability	Patches, New releases, MTTR Database schemas updates
Testability	Support stress tests! Do any special test hooks (pages or APIs) work under stress?
Adaptability	New technology - does weakest point change?
Expandability	Does addition of new services impact capacity?
Re-usability	Re-using stress test? Is reused component weakest link?
Portability	Moving to different servers, services
Interoperability	Weakness of independent servers interoperating!

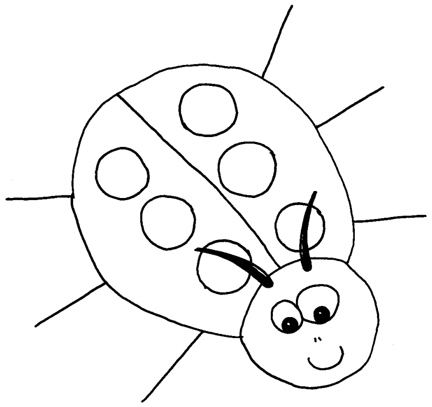


Measurements



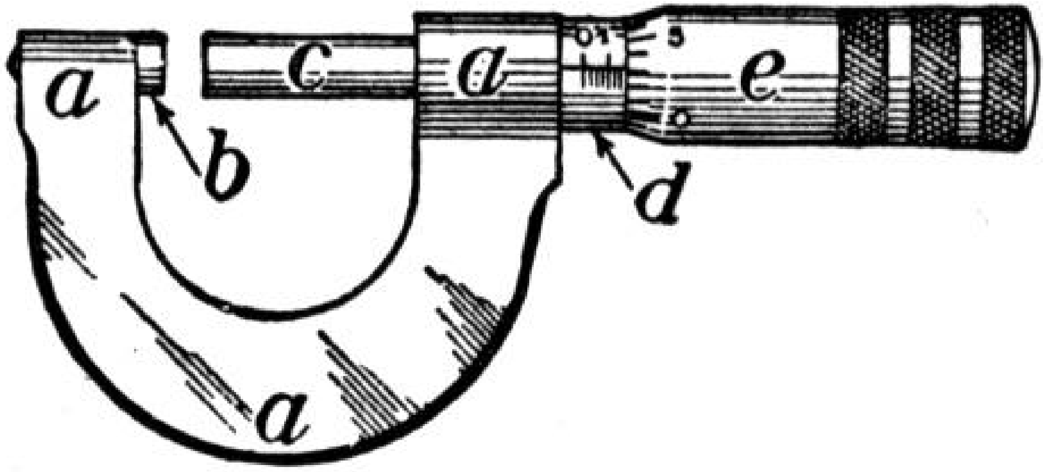
- Response time
  - Minimum
  - Maximum
  - Average
- CPU Usage
- Memory
  - available
  - page faults/second
- Disk
  - % Disk Full

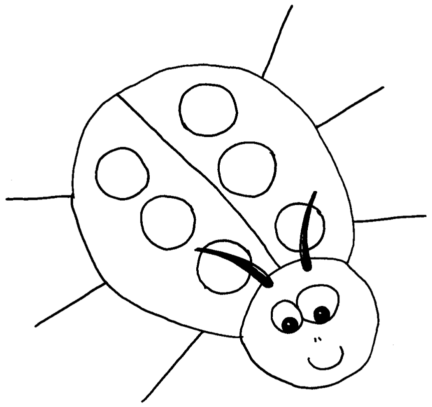




Measurements

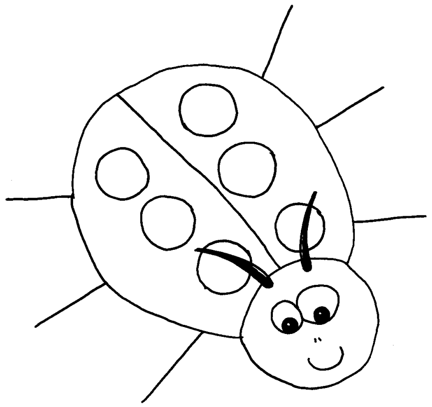
- Network
  - Bandwidth
- Web Servers
  - Files/Sec
  - Bytes/Sec
  - Maximum Connections
  - Errors





Measurements

- Database Servers
  - Transactions/Second
  - Cache hit ratios
- Functionality
  - Pass?
  - Fail?
  - Relation to load



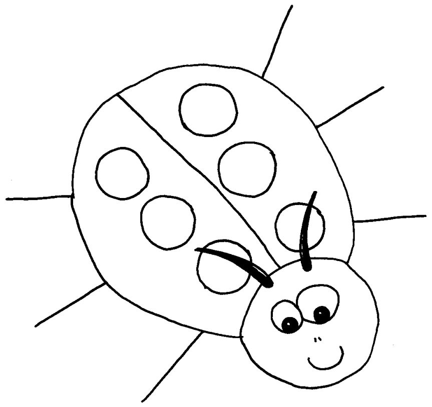
Analysis & Design

Quebec City Bridge, 1916

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

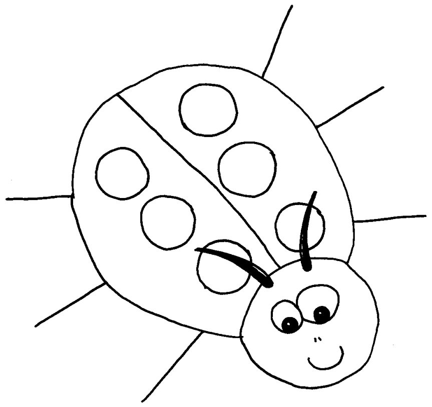
Tacoma Narrows, 1940

- *Collapse during normal operation*
- *Stress due to instability*
- *<u>Wind was only 42 mph</u>*



Computer Simulation

- Computer simulation is used to study how a design, or architecture, will react to stress!
  - Model typical transactions
  - Model atypical transactions
  - Model harsh transactions
  - What if analysis!



Network Simulators

- Many products on the market
- Used by
  - System architects
  - Software designers
  - Network designers
- Test early
  - *Before code is written*

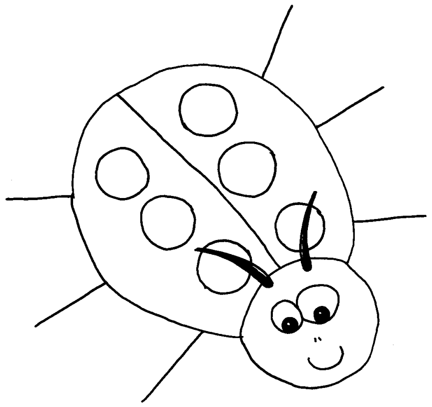
Network Simulator Features

www.nwc.com

	Analytical Engines NetRule 2.3	Compuware EcoPredictor 3.0	NetCracker Technology NetCracker Professional 3.2	Opnet Technologies IT DecisionGuru 6.0
Type of simulation	Analytical and discrete	Analytical	Analytical, discrete for animation	Analytical and discrete
Traffic loading	Messages/hour, message size, reply size, CPU ms, disk ms	Packet rate	Packet size, packet rate	Session rate, packet size, packet rate, response size
Server load	●	❍	❍	Via number of packets processed
Background traffic	●	●	●	●
Load-balancing	●	●	●	●
Routed protocols supported	BGP, IGRP, OSPF, RIP, EIGRP, IS-IS, optimal	IGRP, IS-IS, OSPF, RIP	RIP, IGRP, OSPF	BGP, IGRP, OSPF, RIP
Cost accounting	Cost based on proportional use of links, servers and user wait time	Tariff costs	Equipment list	Equipment list
Scheduling simulation	❍	❍	❍	●
Change control	❍	❍	❍	●
Web reports	❍	●	●	●
Topology import	ASCII text file	Compuware EcoScope, CA Unicenter, HP OpenView, Tivoli NetView	HP Network Node Manager, Visio	HP Network Node Manager, Tivoli NetView, Excel
Traffic Import	ASCII text file	Compuware EcoScope, Network Associates Sniffer Pro, NetScout Systems NetScout	❍	HP NetMetrix, NetScout, Expert Sniffer, six ASCII formats
Tutorial	●	●	●	●
Single-unit retail price	$7,500	$24, 500	$9,995	$19,000 plus $7,000 for MVI and $9,000 for ESP modules
Warranty	One year	90 days	30 days	90 days
Ar[illegible]ai[illegible]	[illegible]% of list	12% of list	24% of list	18% of list
Ma[illegible]c[illegible]e[illegible]	[illegible] phone [illegible] e-mail technical s[illegible]ort during business [illegible]iness day [illegible]pgrades	Software updates, documentation updates, [illegible]	Unlimited technical support 24x7, monthly database updates and [illegible]	Unlimited technical support, core tool updates, major release, model [illegible]y updates, access to our [illegible]mmunity through our Web site, access to Client Consultation and Computing Center

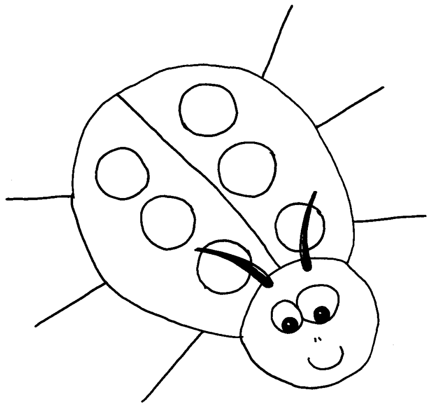
Network Computing

THE TECHNOLOGY SOLUTION CENTER



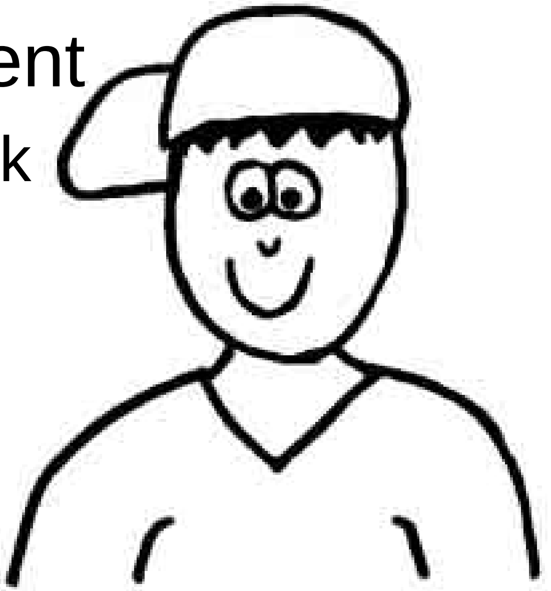
Performance Model

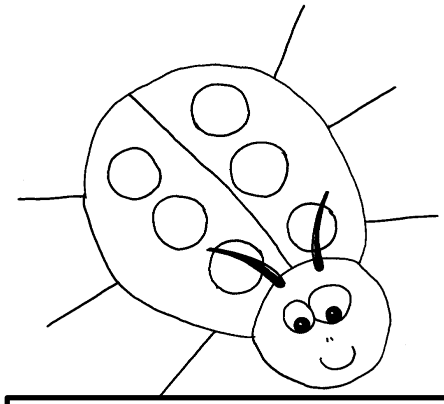
- SPE-ED
  - Software Performance Engineering Performance Modeling Tool
  - *"…tool that produces performance data on development alternatives without requiring extensive knowledge of modeling theory…"*
  - www.perfeng.com



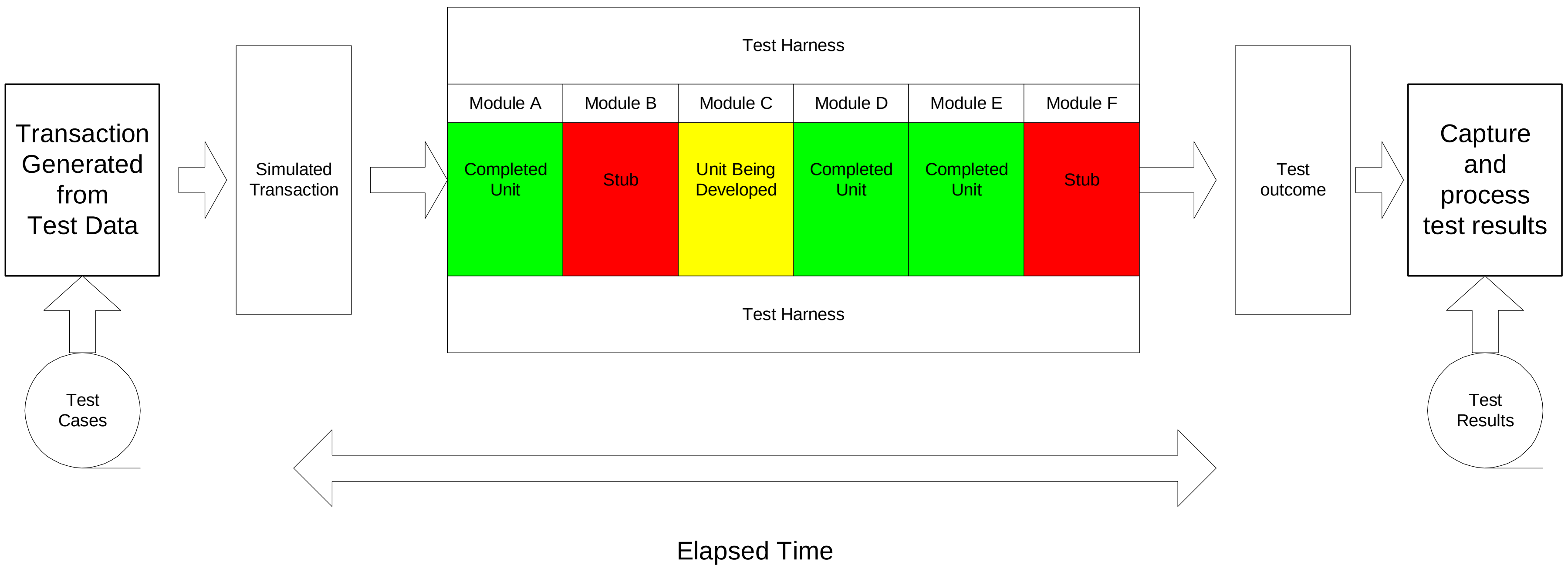
Development

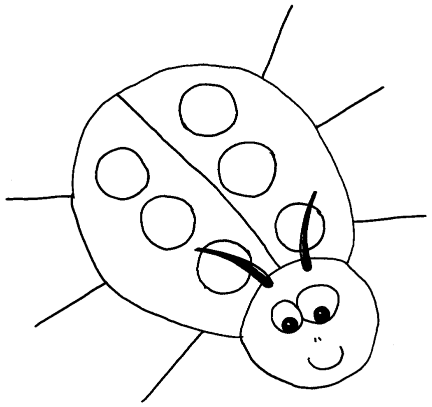
- Include stress testing during unit testing!
  - Before and after task assignment
    - Develop test cases as part of task
      - XP’ish is good
    - Does system behave the same?
    - Differences as expected?
  - Fewer surprises later in cycle.
  - Work with testing team to build harness





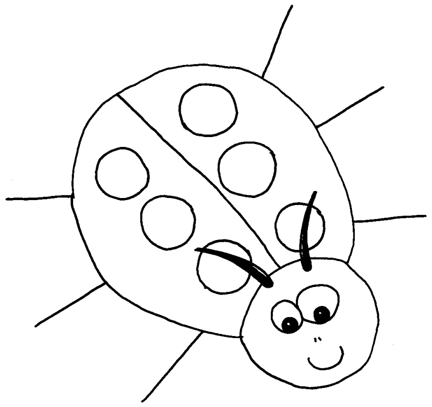
Unit Test Harness





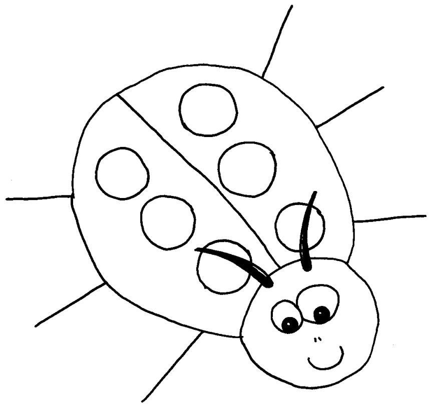
Testability Hooks

- It is wise to ensure that testability is considered in all code developed!
  - Simple static page to access each business logic function independent of GUI.
  - Runtime enable and disable of logging features. *(web, business logic, data tier)*
  - Access to key objects.
  - Hooks to facilitate measures!

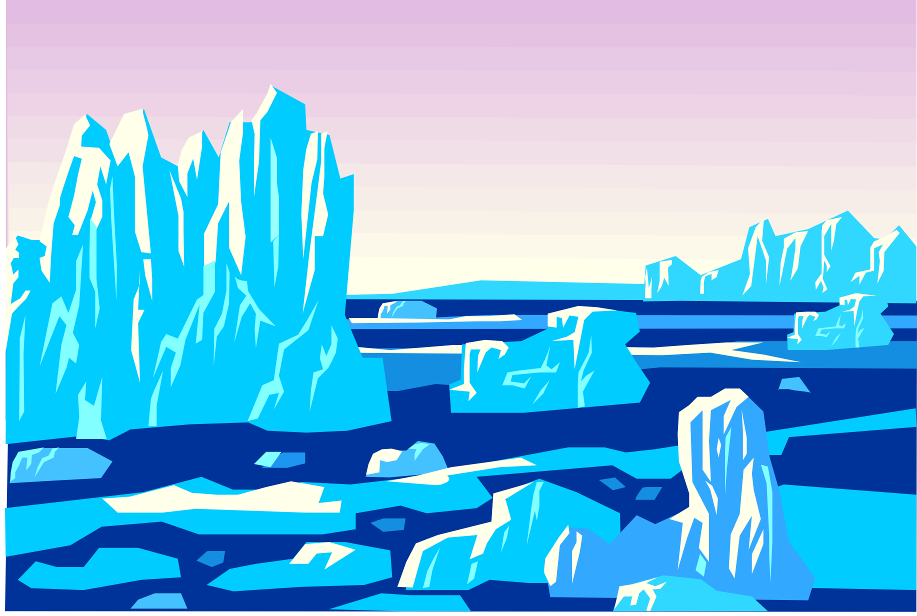


Approaches

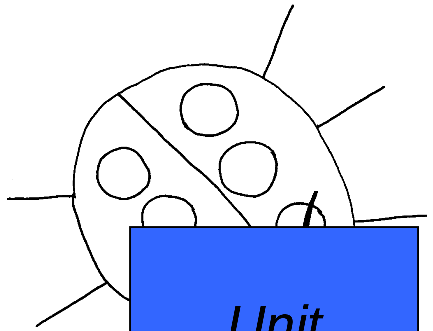
- Peer review on check-in
- Formal inspections especially focused on identifying weak links in design or any problems with architecture or requirements
- Unit testing to include Stress over and above Functional testing



Issues

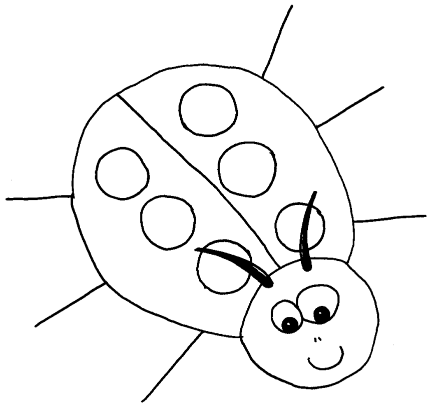


- Watch out for
  - Moving instabilities
  - Changes to database schemas for new features
  - Developers unfamiliar with code base
  - Reusing code “blindly” to save time
  - Increasing complexity of code
  - Differences between target and developer “Run-time” environments



Testing Phase

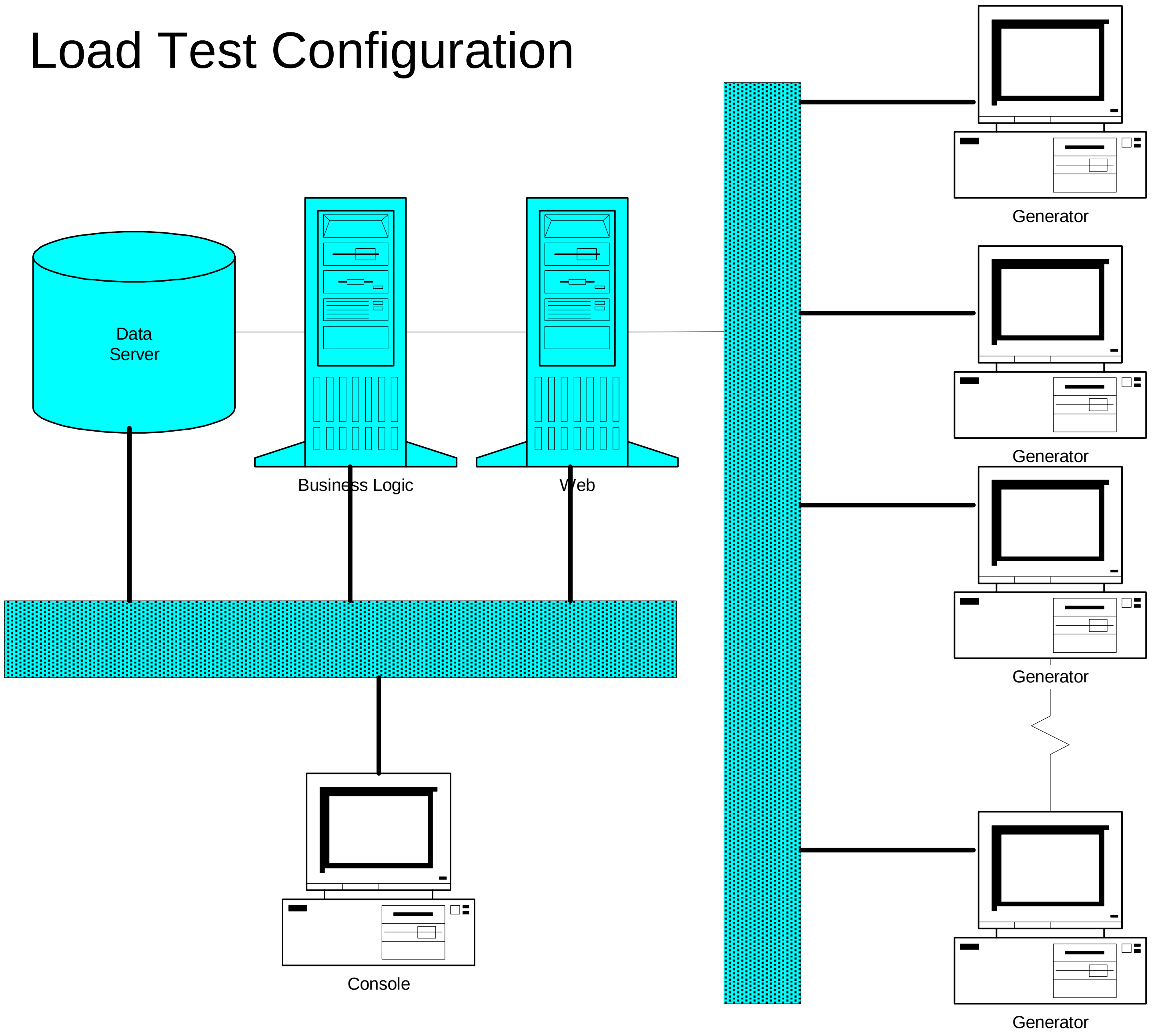
	Who?	*Where?*	*Tools & Techniques*
Unit	Developers	Developers test server Development office	HTTP Event generator Home made tools Client side test tools Server based load
Integration	Developers Independent testers	Test Lab Development configuration	Test automation web Client side load testing Server monitoring
System	Independent testers	Test Lab Staging site matching target site (smaller but similar)	Test automation web Client side load Server monitoring Load testing tools
Acceptance	Client testing testing team	Staging area on Live site	Test automation web Client side load testing Server monitoring Load testing tools
Live	Independent test monitor	Live site	Load testing services Site monitoring services Internal monitoring and periodic testing

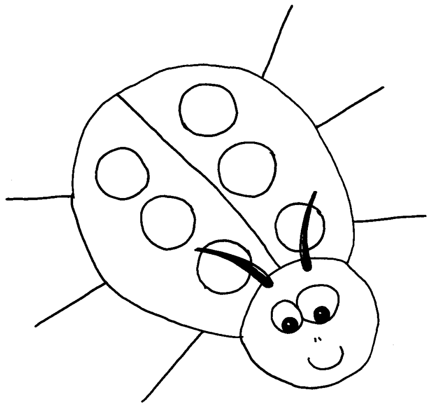


Stress Testing Environment

- N-tier setup
- Generally
  - Client
  - Application
  - Data

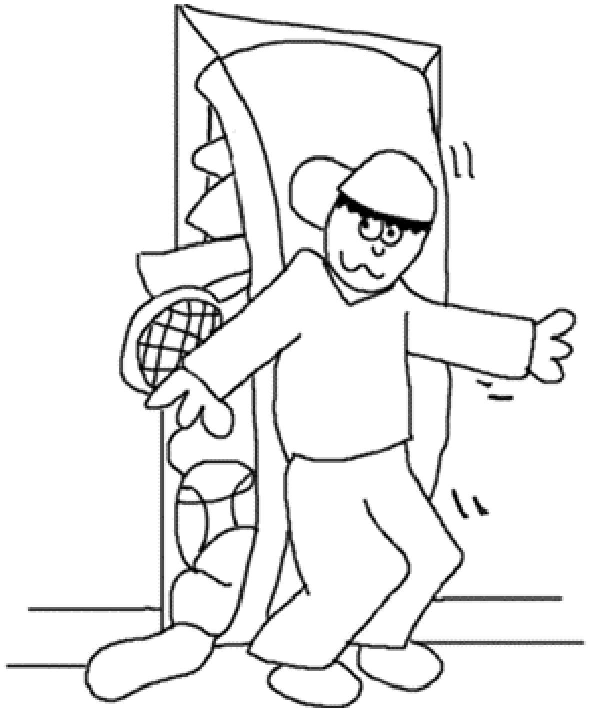
Load Test Configuration



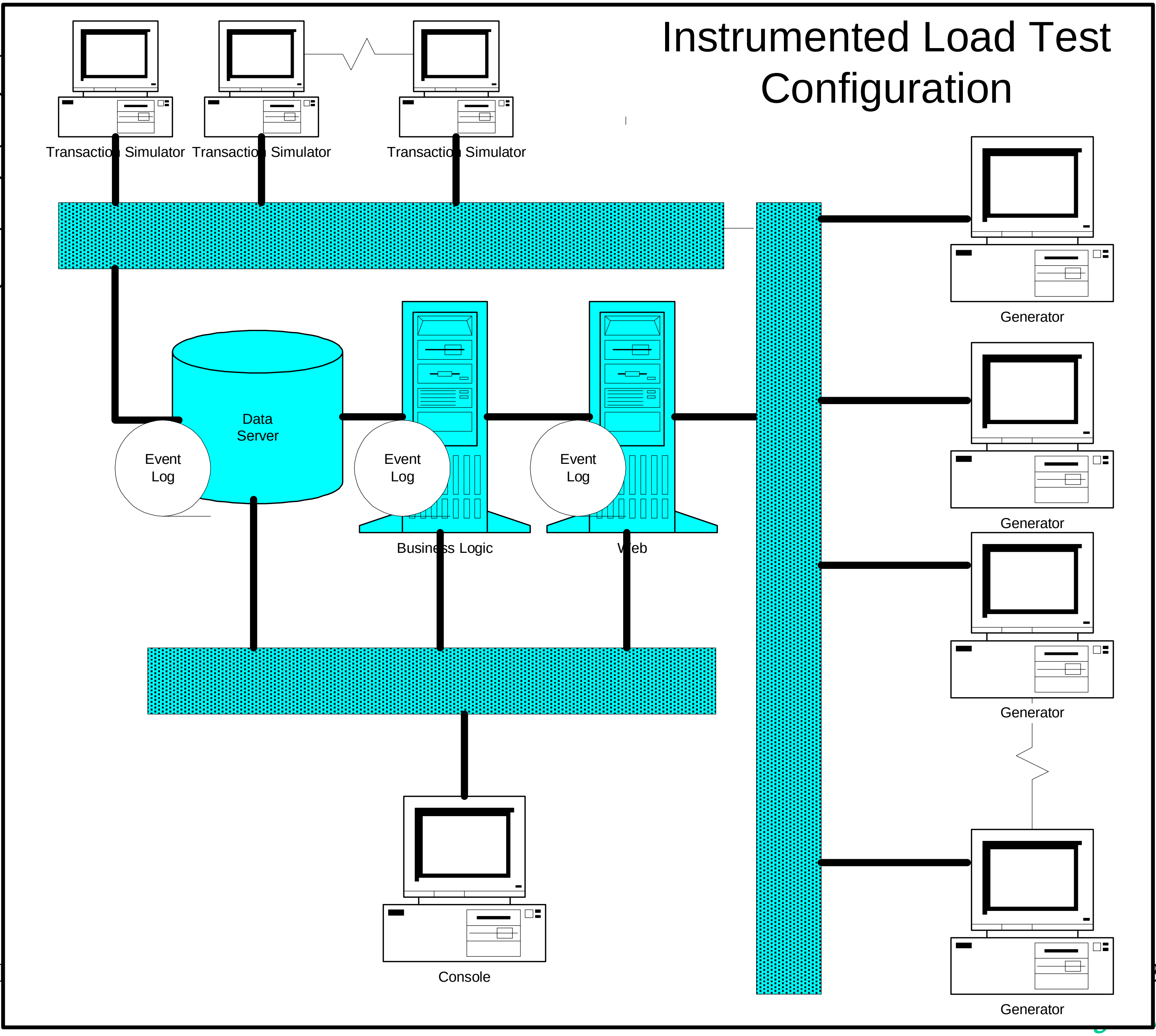


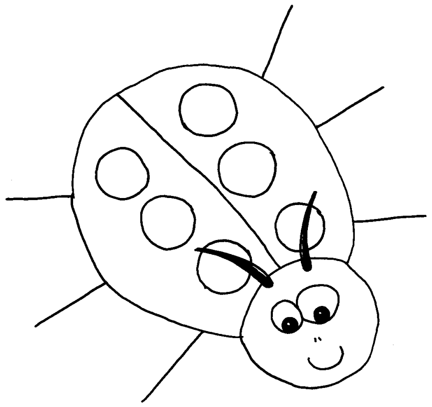
Load Testing

- Load generator
  - Keep the system busy!
  - Simulate of user experiences
  - Several concurrent “Virtual users”
  - Generally you should model most common type of user experiences
  - Experiences can be:
    - TYPICAL
    - HARSH



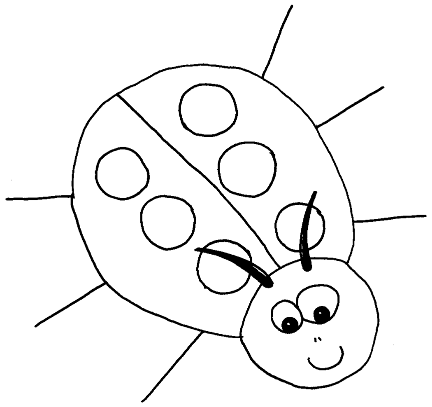
Instrumented Load Test Configuration





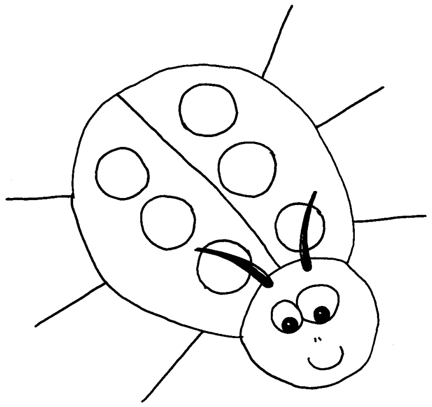
Transaction Simulator

- Typical user transactions
- Measure how well the system performs these transactions as we vary the load
  - Correct?
  - Time?



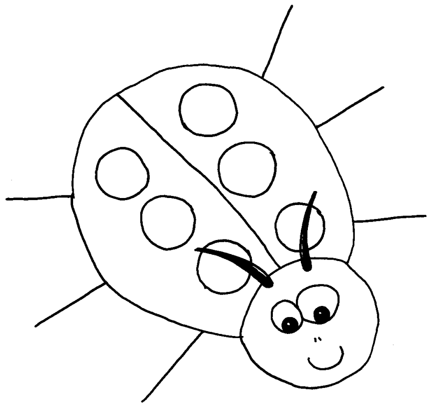
Transaction Simulator

- For example:
  - Purchase scenario is run on the transaction simulator
  - We generate a load on the system with the load generators and study how this impacts the typical transactions
  - Load generator is focused on keeping the system busy!



Performance Measures

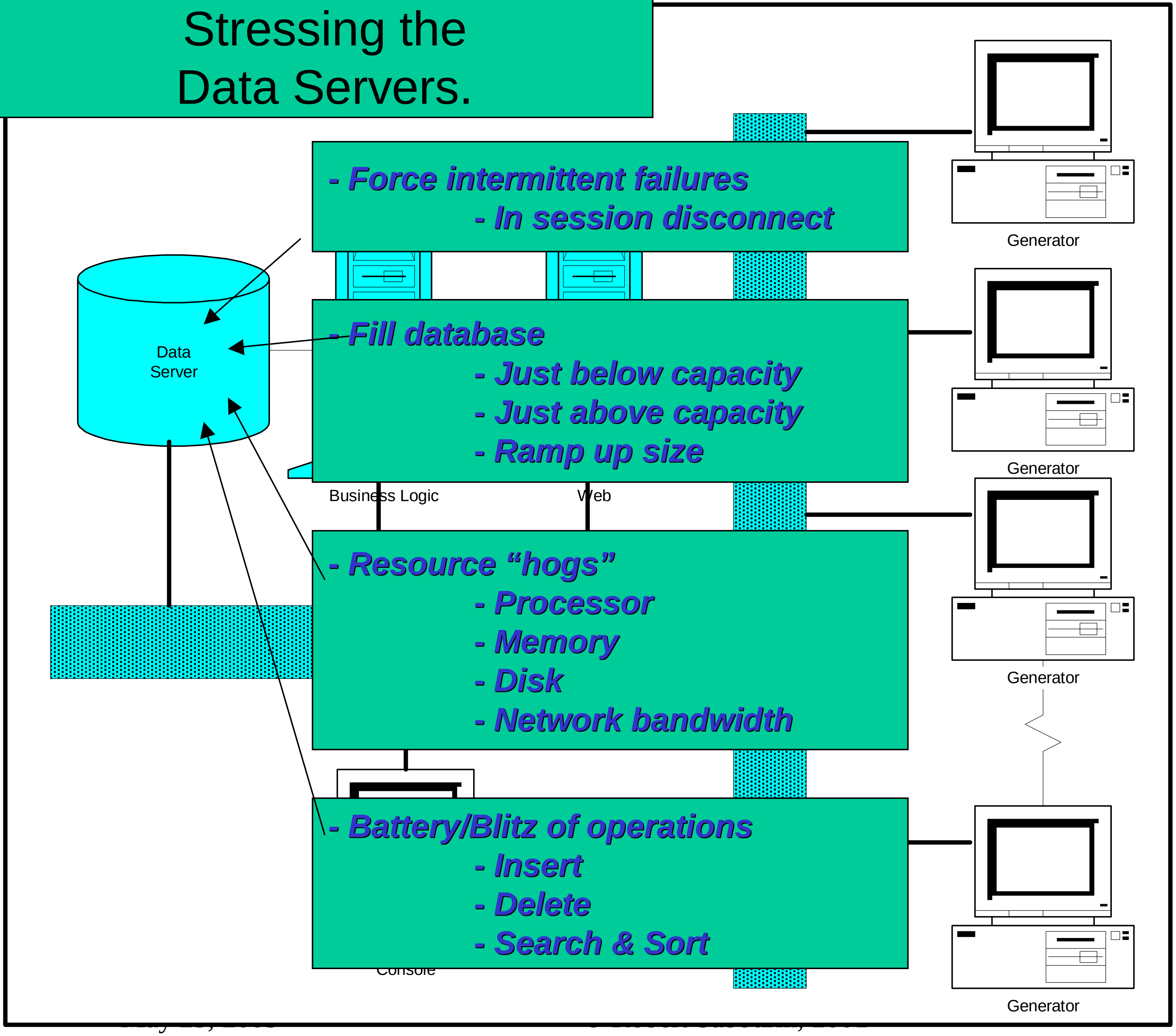
- User perspective
  - Time to complete a transaction from the user perspective
  - Time to get a responses to an input event
    - Should be maximum 8-10 seconds
- Page load
  - time to last byte
- “Response time” of the system
  - time from a mouse click until the next page is finished loading



Performance Measures

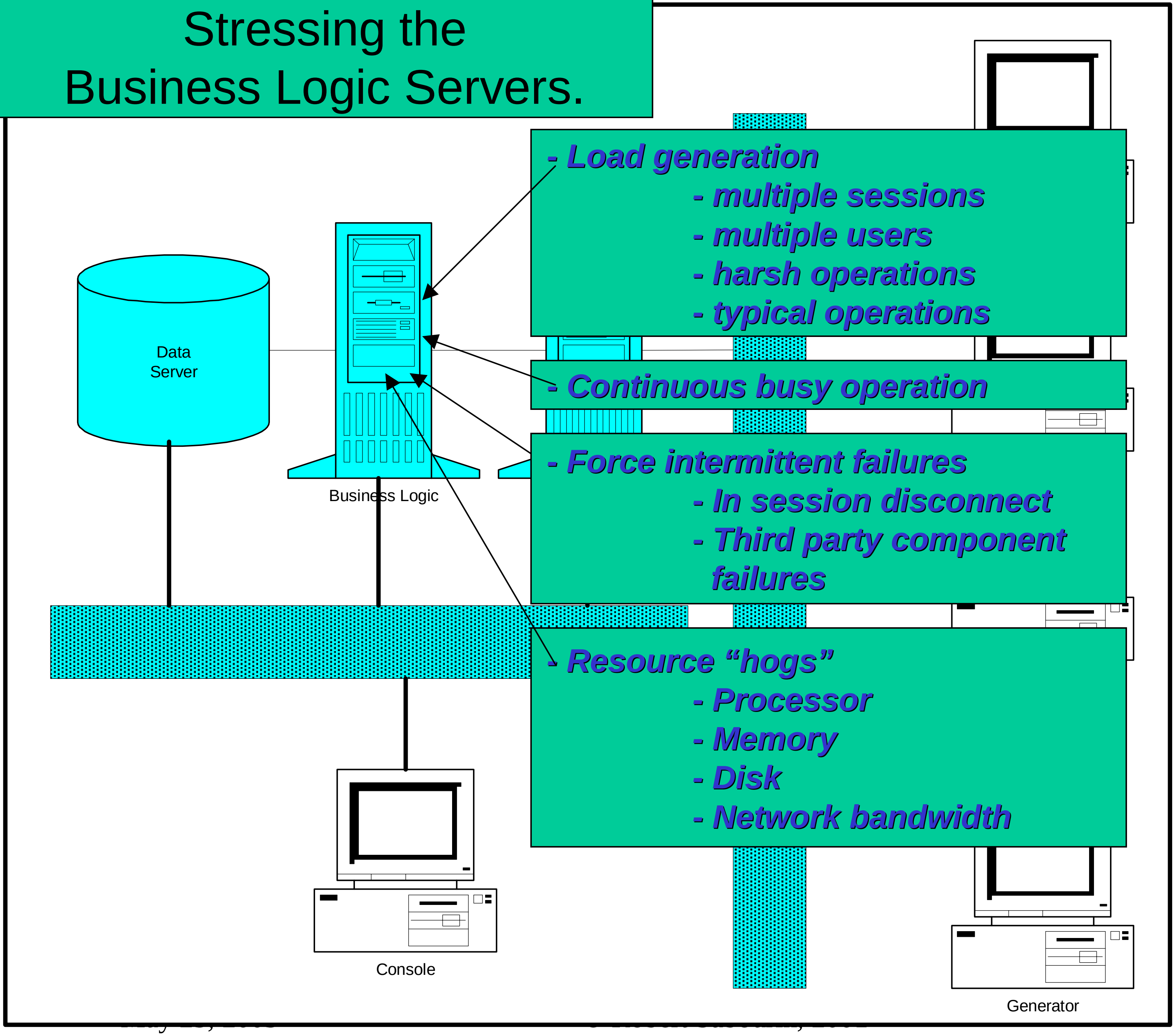
- Components of "Response time"
  - – Request Submission
    - Data to server
  - – Processing Time
    - Time spent working on user request
  - – Response Receipt
    - Time to send result back to user

Stressing the Data Servers.



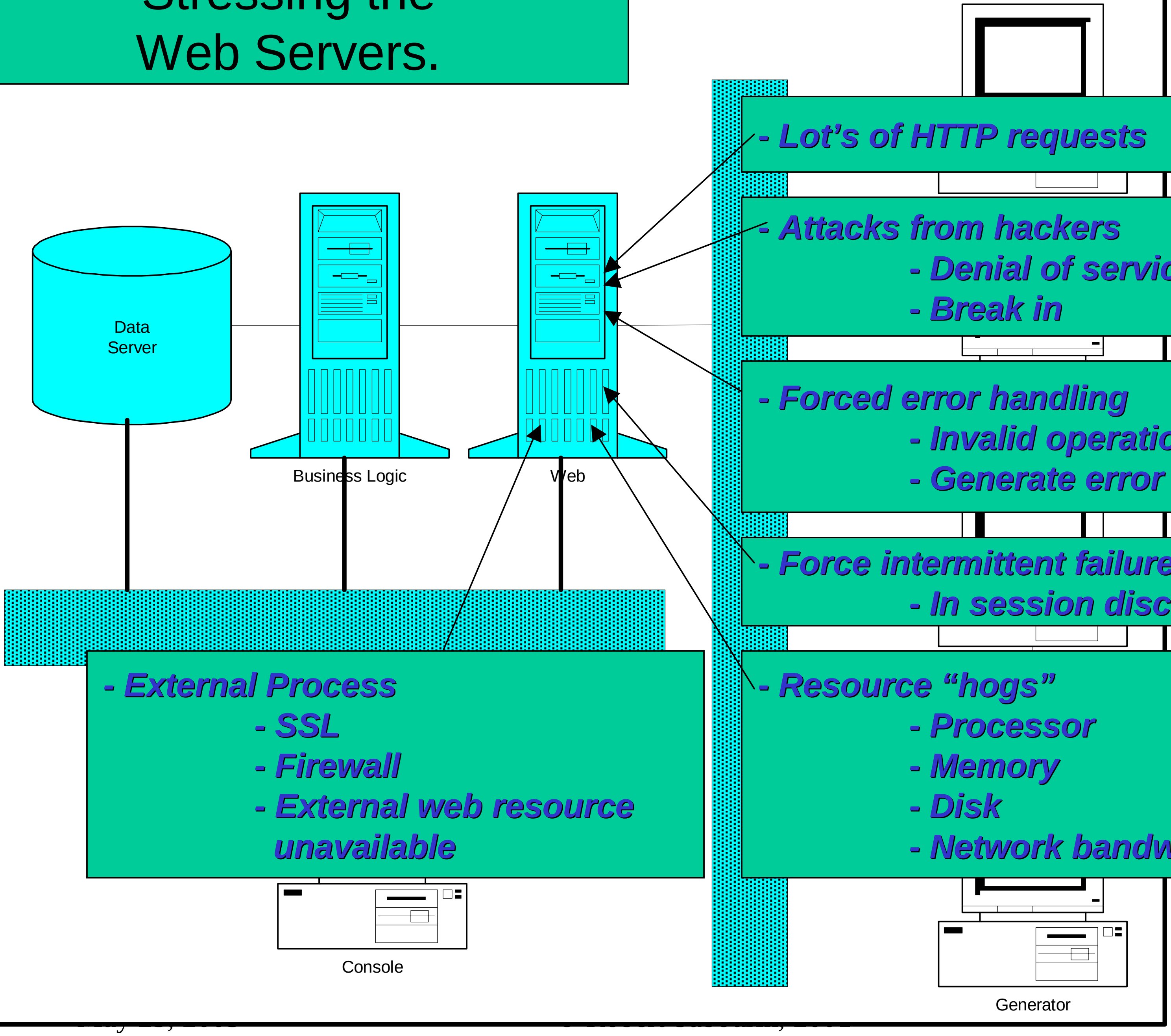
- *Force intermittent failures*
  - *In session disconnect*
- *Fill database*
  - *Just below capacity*
  - *Just above capacity*
  - *Ramp up size*
- *Resource "hogs"*
  - *Processor*
  - *Memory*
  - *Disk*
  - *Network bandwidth*
- *Battery/Blitz of operations*
  - *Insert*
  - *Delete*
  - *Search & Sort*

Stressing the Business Logic Servers.



- Load generation
  - multiple sessions
  - multiple users
  - harsh operations
  - typical operations
- Continuous busy operation
- Force intermittent failures
  - In session disconnect
  - Third party component failures
- Resource "hogs"
  - Processor
  - Memory
  - Disk
  - Network bandwidth

Stressing the Web Servers.



- Lot's of HTTP requests

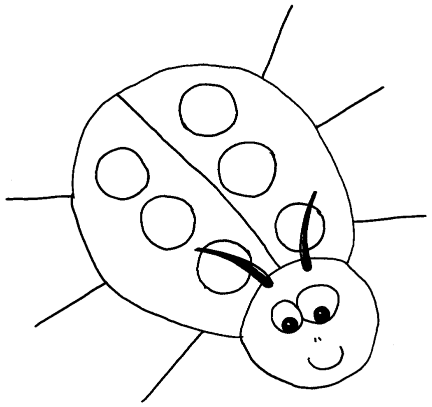
- Attacks from hackers
  - Denial of service
  - Break in

- Forced error handling
  - Invalid operations
  - Generate error conditions

- Force intermittent failures
  - In session disconnect

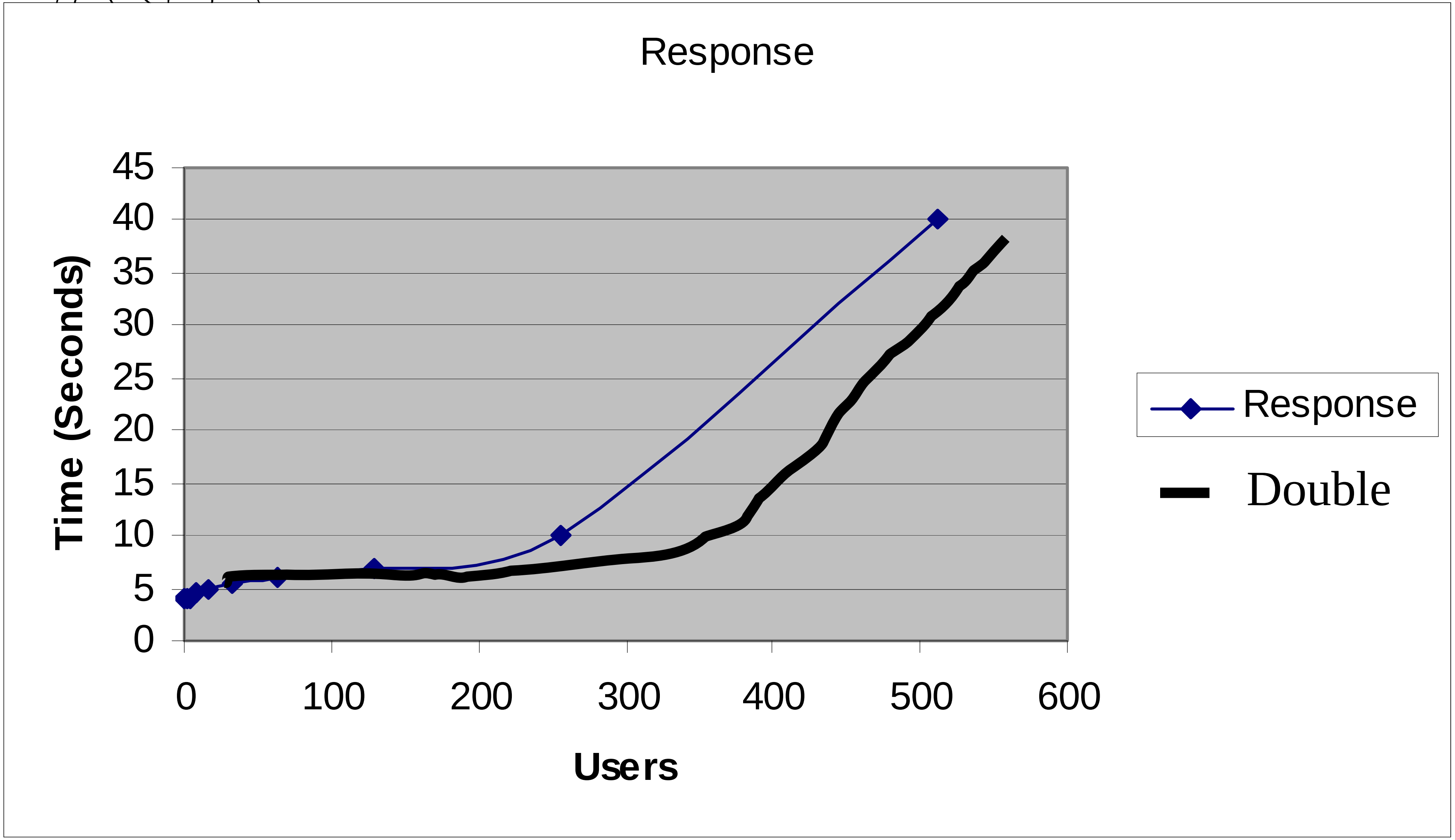
- Resource "hogs"
  - Processor
  - Memory
  - Disk
  - Network bandwidth

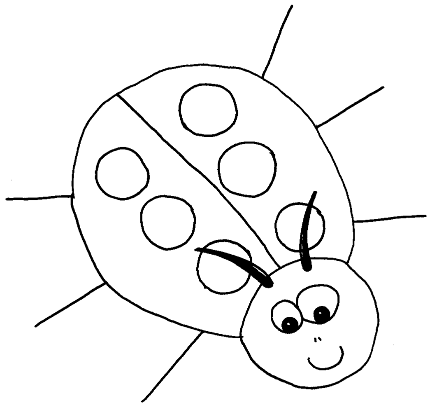
- External Process
  - SSL
  - Firewall
  - External web resource unavailable



Scalability

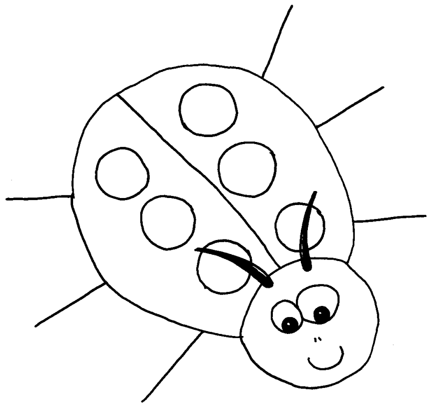
- Vertical
  - Replace servers with more powerful systems
  - Add resources to existing servers
- Horizontal
  - Add more servers to the site
  - Needs load balancing technology
  - Increases availability of site
- Functional
  - Separate application functions onto different servers and scale them independently


Response
Time (Seconds)
45
40
35
30
25
20
15
10
5
0
0
100
200
300
400
500
600
Users
Response
Double



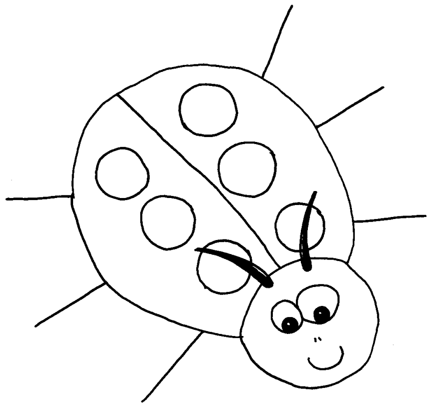
Stress Test Planning

- Time
  - Generally there is not enough time to do exhaustive stress testing of all components of the system
- Risk
  - Spreading resources across different stress testing activities must be done carefully based on the technical risk and potential business impact of failures



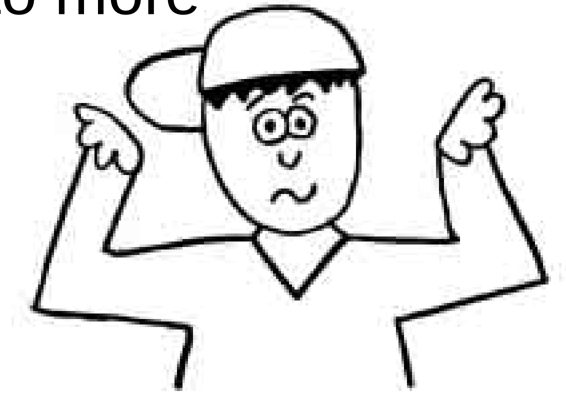
Stress Test Planning

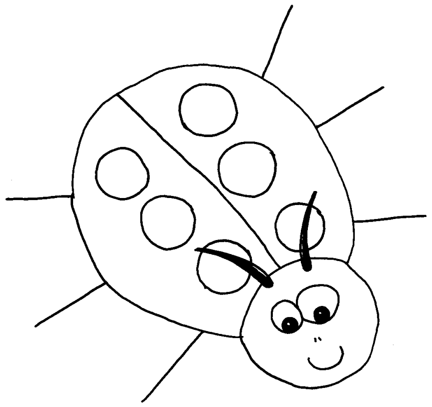
- Technical risk
  - New code
  - Old code used in a new w
  - New developer
  - New hardware
  - New third party stuff
  - High risk
  - Complexity



Stress Test Planning

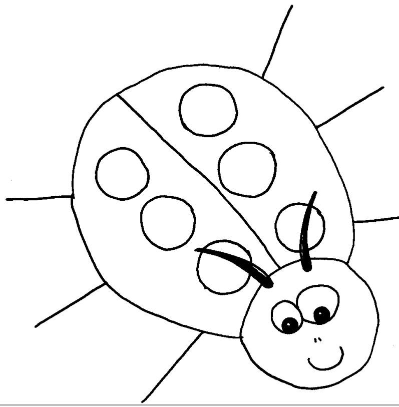
- Business Impact
  - Impact of performance degradation?
  - Impact of missing functionality?
  - Load balancing?
  - What about diverting CPU power to more popular functions and disabling less popular operations?





Stress Test Planning

- Pattern Evolution:
  - List relevant stress tests
  - *Guestimate* risks working with peers in product management, development and other stakeholders
  - Spread testing across builds relative to risk from highest to lowest
  - Try to implement at least *one Stress Test experiment per build*.

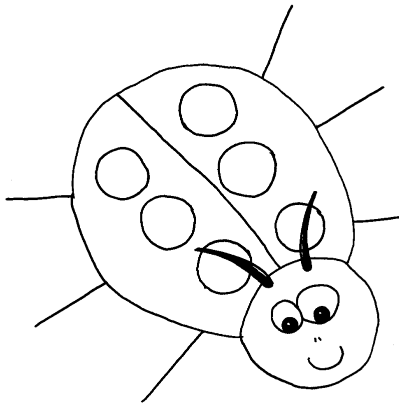


Stress Test Experiments

Example Stress Experiment

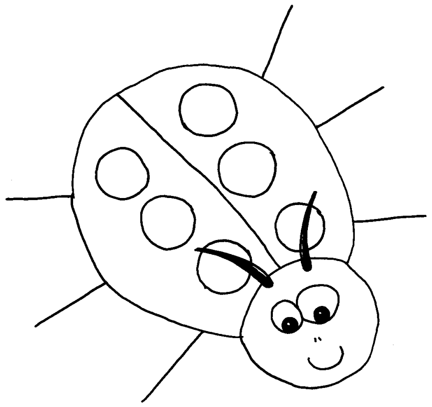
Concurrent Users	Load Generated	Database State	Technical Risk 1 to 10	Business Importance 1 to 10	Exposure	Rank	Build A	Build B	Build C	Build D	Build E	Build F
1-1000	Normal	Low	3	10	30	3			🟨			
1-1000	Harsh	Low	5	5	25	5					🟦	
1-1000	Normal	Medium	3	10	30	4				🟨		
1-1000	Harsh	Medium	5	5	25	6						🟦
1-1000	Normal	High	8	8	64	1	🟥					
1-1000	Harsh	High	10	5	50	2		🟥				

Critical	🟥	Exposure > 50
Serious	🟨	Exposure between 26 and 50
Important	🟦	Exposure between 15 and 25
Nominal	🟪	Exposure less than 15

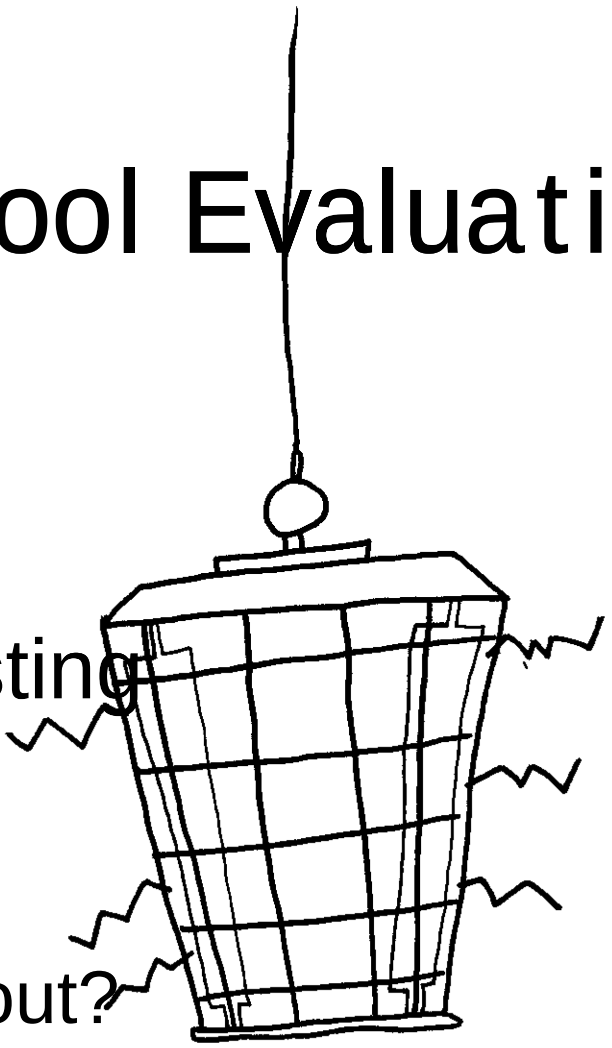


Stress Test Preparation

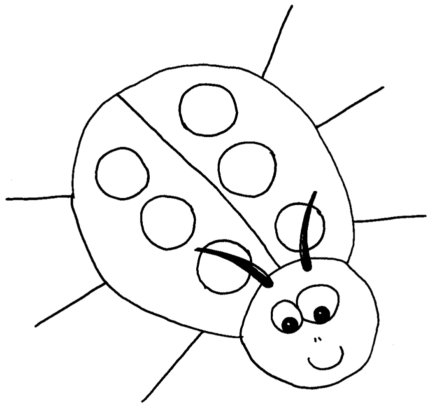
– Warm up, caches and buffers to steady state
– System to desired starting state
– Ensure no one else is accessing system under test
– Ensure databases and system resources are in the correct initial state to start testing



Test Tool Evaluation

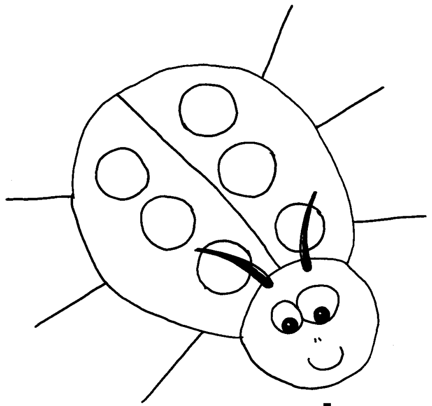


- Tools to help you perform stress testing
  - What to buy?
  - When to buy?
  - What to contract out?
  - What to do yourself?



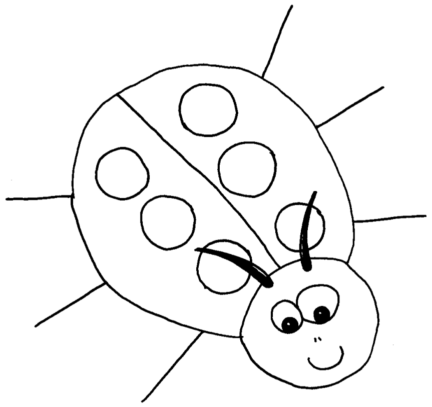
Test Tool Evaluation

- Tools to help you perform stress testing
  - Event Simulation
  - Load Generation
  - Site Monitoring
  - Test Harnesses
  - Environment
    - Monitor
    - Fault injection



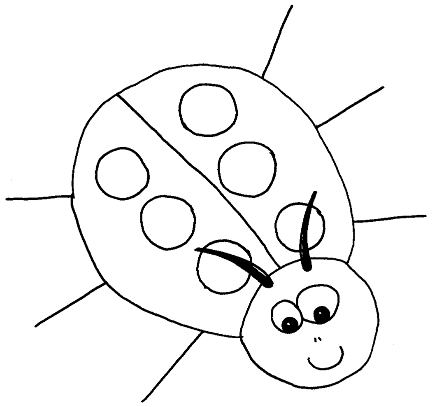
Test Tool Evaluation

- What to buy?
  - – Things you will *reuse* on many projects.
  - – Things which *save you time*
  - – Things which will save you *money*.
  - – Itemize your *specific requirements*.
  - – Can you buy part of the solution now and part later to spread the cost?
  - – Can you leverage internal expertise?
  - – Will significant training be required?



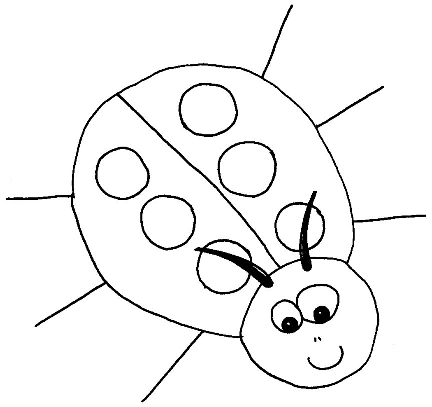
Test Tool Evaluation

- What to buy?
  - Be careful about maintenance, especially for open source or otherwise free tools!
    - Technologies change fast in www
    - Support for third party gizmos and widgets is important.
    - How quickly does your supplier adapt to new technologies?
    - Will you be able to keep your own, in-house, tools up to date?



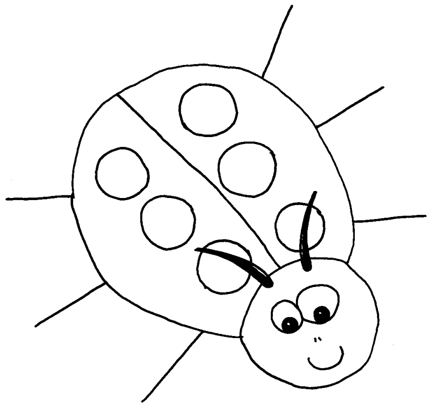
Test Tool Evaluation

- What to buy?
  - What if your requirements change!
    - If you swap between Linux, NT, Solaris will the tools still work?
    - If you move between SQL Server, Oracle, Interbase will monitors still work?
    - If you change http servers from Apache to IIS will tools still work?



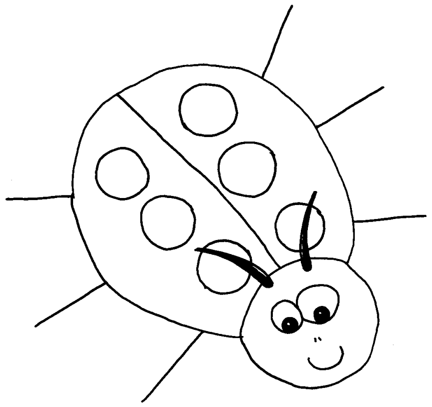
Test Tool Evaluation

- When to buy?
  - Company issues
    - If your company is going to develop several projects using the same or tightly related technologies then it is wise to tool a test lab independent of a specific project schedule

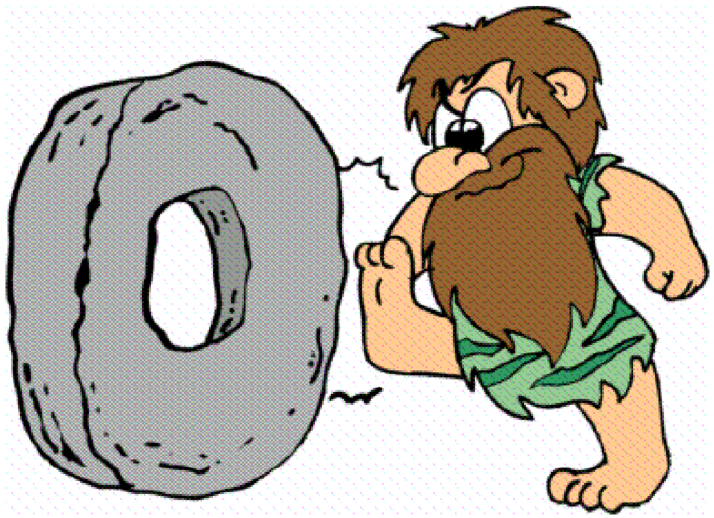


Test Tool Evaluation

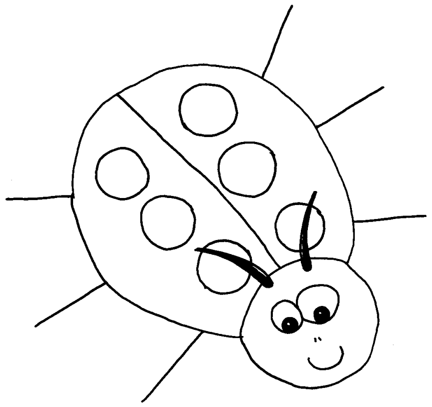
- When to buy?
  - Project issues
    - Identify vendors during early phases
    - Use first iterations to evaluate tools
    - Buy tools before full system testing
    - You should have some sort of load generator during Unit, Integration and System Testing Phases



Test Tool Evaluation

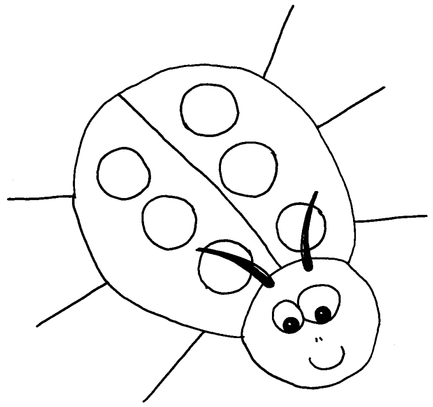


- When to buy?
  - After evaluation.
  - TRY BEFORE YOU BUY.
  - Be honest with vendors and tell them what type of evaluation process you have decided to use.
  - Let them know that other suppliers are also being considered.
  - Ensure they help you get stuff - really - working

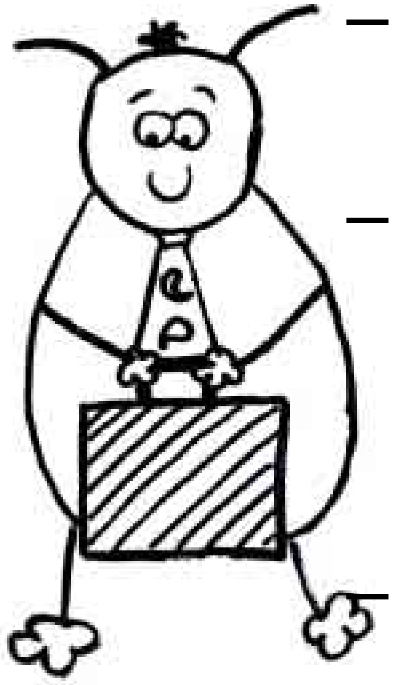


Test Tool Evaluation

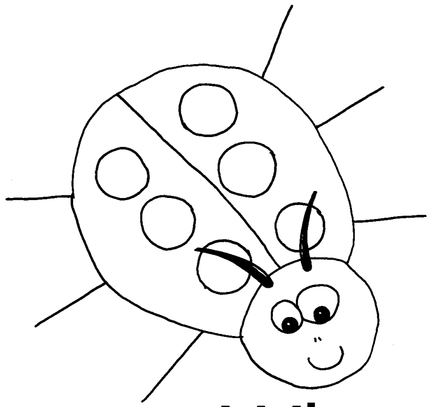
- What to contract out?
  - Things you are only doing once!
  - Things that you do not have the expertise for in-house! *(first time load scripts)*
  - Things that require investments you are not ready to take yet.
  - Things that will save you time.
  - Things that allow more parallelism in development.



Test Tool Evaluation

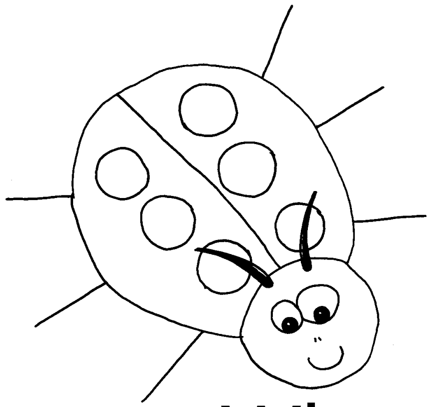


- Consultants?
  - Mapping out an effective strategy to achieve your business goals.
  - Offer guidance, coaching, mentoring and management consulting from someone who as been through the experience before.
  - Each case is different!



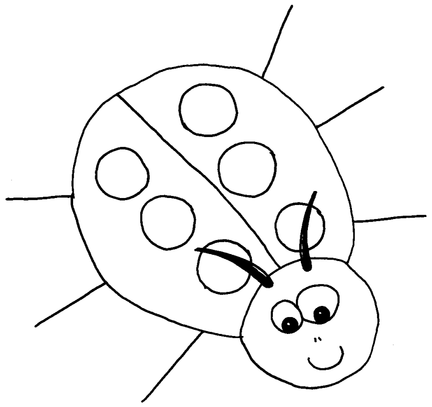
Test Tool Evaluation

- What to do yourself?
  - Unit Test Harnesses
    - Your test and development team know your application and development environment best
    - These tools must change as the project is developed
  - Test Hooks
    - Put in special test access points
    - Simple HTML forms instead of animation



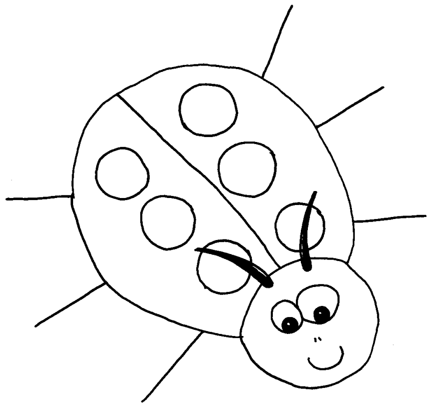
Test Tool Evaluation

- What to do yourself?
  - Monitors
    - Snap shots of system status
    - View system parameters of special interest
    - Special log file analyzers
  - What are you expert in?
    - If you have some expertise available then use it! Just make sure that you don’t just hack a tool together and forget about maintaining and evolving it as your needs change down stream!



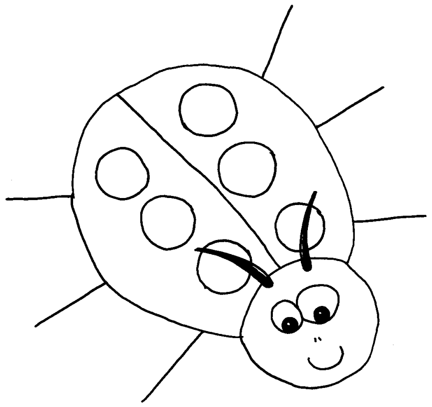
Home Brew

- Monitors from Perl/Shell Scripts on Unix Box
    - CPU status
    - Virtual Memory Status
    - Available Disk Space
    - Available DBMS Space
    - Disk Usage
    - DBMS Usage
    - Number of concurrent users
    - Process status
    - Log file analysis
    - Spreadsheets and Excel Macros!
    - Database for tracking stress testing experiments



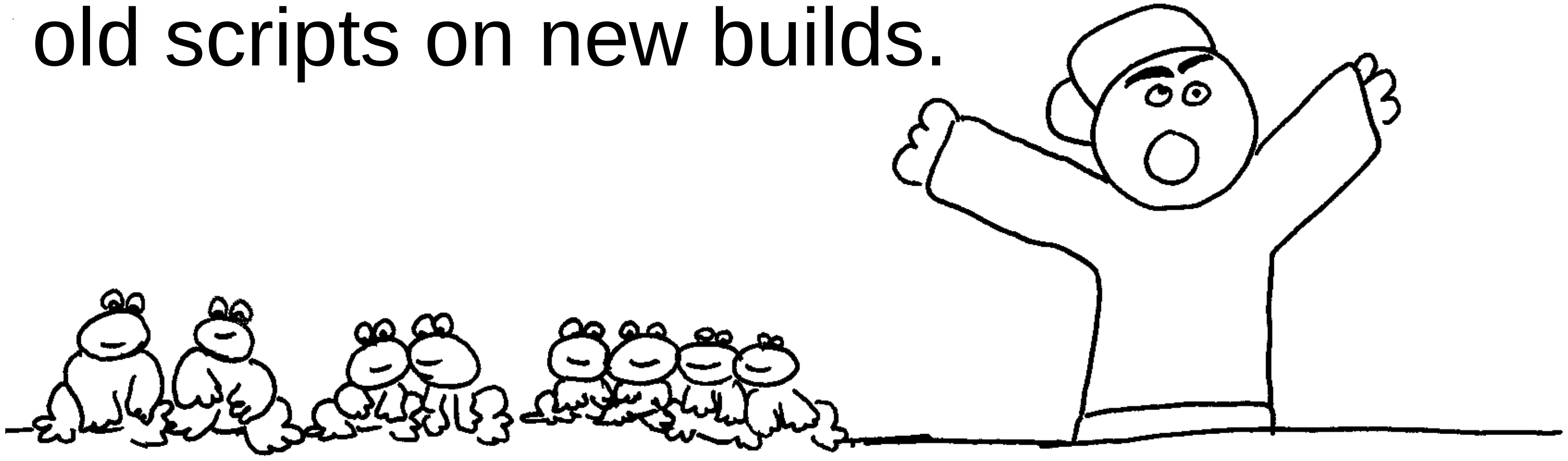
Testing Services

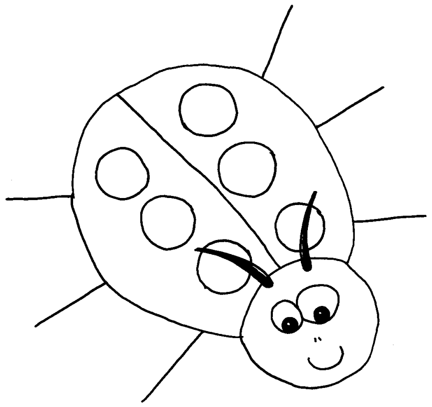
- Load testing
  - Perform load testing
  - Price generally related to volume
- Site monitoring
  - Remote monitoring and measurement of site performance
- Contract testing
  - Outsource testing to experts



Real World Frustration

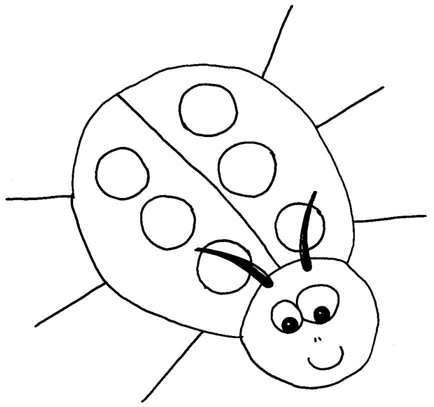
- Often, during stress testing, applications fail while you are trying to develop test scripts, procedures or trying to debug old scripts on new builds.



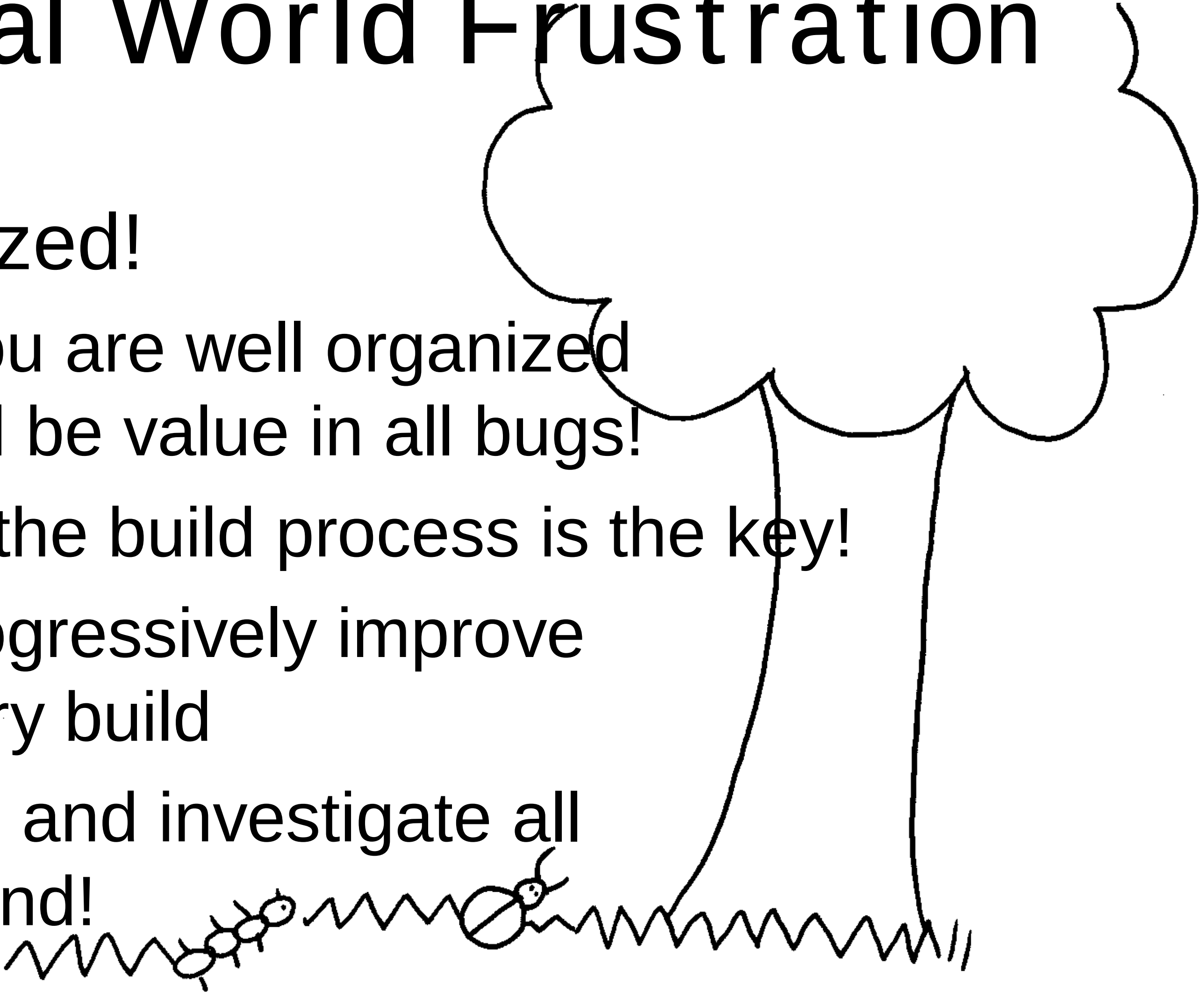


Real World Frustration

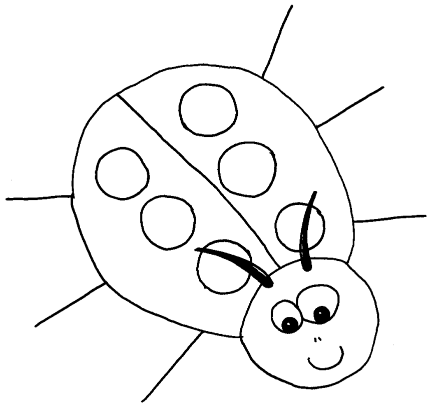
- Bugs got you down!
  - Be patient!
  - Focus on stressing application not the testing team!
  - You will find bugs with stress testing because you are stress testing!
  - Sometimes the bug is in the test and sometimes the bug is in the program being test.



Real World Frustration



- Be organized!
  - When you are well organized there will be value in all bugs!
  - Rigor in the build process is the key!
  - Build progressively improve with every build
  - Prioritize and investigate all bugs found!



Thank You

- Questions?