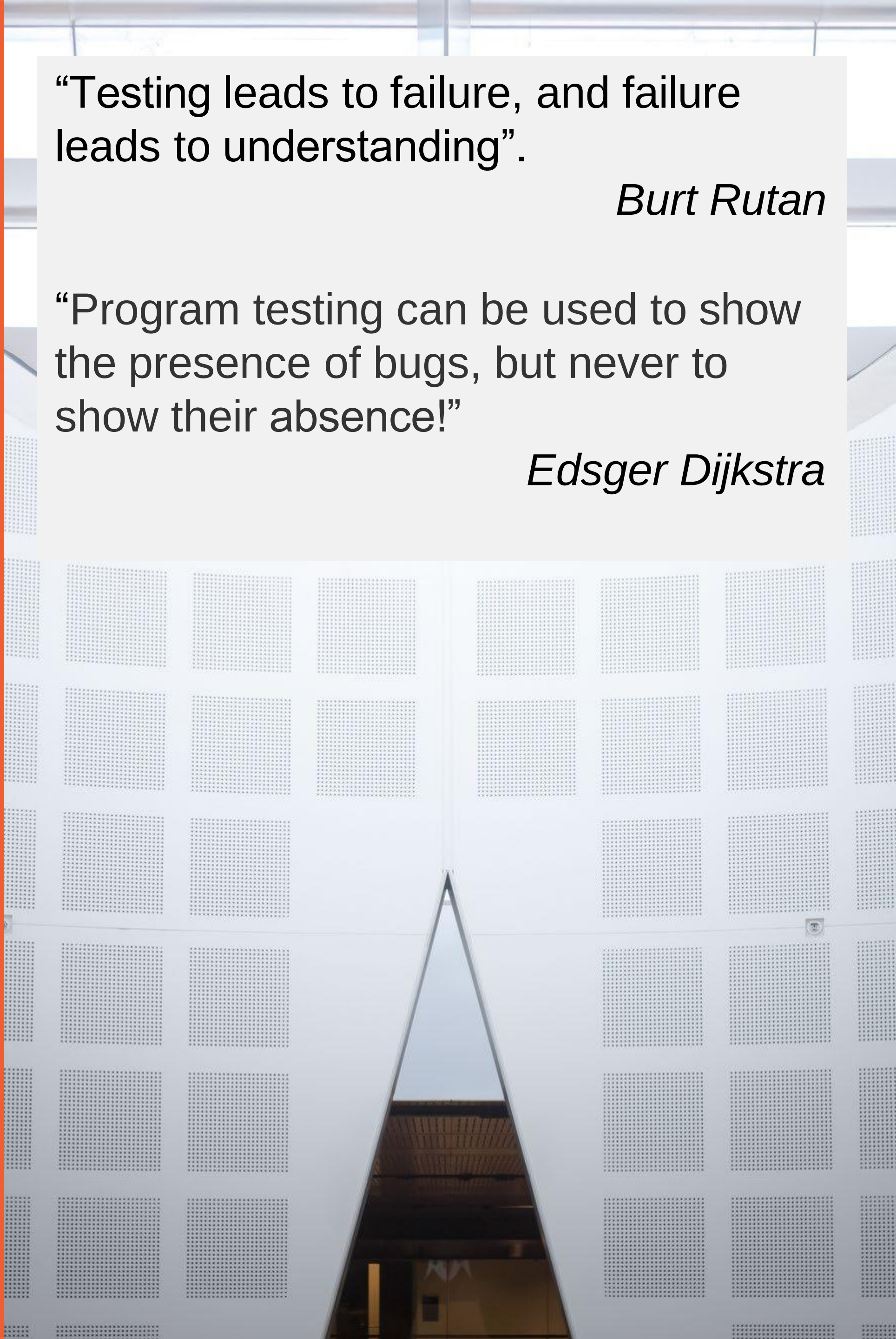
INFO5990: Professional Practice in IT

Week 7: Test Management

School of Computer Science

THE UNIVERSITY OF SYDNEY



"Testing leads to failure, and failure leads to understanding".

Burt Rutan

"Program testing can be used to show the presence of bugs, but never to show their absence!"

Edsger Dijkstra

Software Testing

- A systematic process of evaluating software to:
    - Confirm it meets specified requirements (Verification).
    - Ensure it satisfies user needs (Validation).
    - Identify defects, bugs, or issues.
    - Assure quality and mitigate risks.
    - Evaluate performance, security, and usability.

Testing Phases

Requirements Phase:	• **Requirements Review:** Testing begins by reviewing and validating the software requirements to ensure they are clear, complete, and achievable.
Planning Phase:	• **Test Planning:** Detailed test plans are created, outlining testing strategies, objectives, resources, and schedules.
Design Phase:	• **Test Design:** Test cases and test scenarios are designed based on the software design and requirements.
Development Phase:	• **Unit Testing:** Developers conduct unit testing to evaluate individual code units (e.g., functions or modules).
Integration Phase:	• **Integration Testing:** Testing focuses on interactions between integrated components or modules.
System Phase:	• **System Testing:** The entire software system is tested to ensure it functions correctly as a whole.
Acceptance Phase:	• **User Acceptance Testing (UAT):** End-users test the software to verify it meets their needs and expectations.
Release Phase:	• **Deployment Testing:** Testing is performed to ensure a smooth deployment of the software in the production environment.
Maintenance Phase:	• **Regression Testing:** Tests are rerun to verify that new changes do not introduce new defects.

Testing Phases Example

Example: Library Management System

The system allows:

- Students to search for books
- Borrow or return books
- View due dates
- Admins to manage inventory

1. **Requirements Phase**

   Activity:

   Gather features like:

   - Students must log in to borrow books.
   - System should show available copies and due dates.

   Testing:

   Requirements Review: Check that requirements are clear and make sense for both students and staff.

Testing Phases Example

2. Planning Phase

Activity:

Plan the testing activities:

•What will be tested?

•Who will test it?

•What tools or data are needed?

Testing:

Test Planning: Create a schedule, assign roles, and list test items (e.g., login, borrow feature, return logic).

3. Design Phase

Activity:

Design the database, system flow (e.g., student → search book → borrow), and user interface.

Testing:

Test Design

Create test cases like:

- "If a book is unavailable, user should not be able to borrow."
- "Due date should be calculated correctly."

Testing Phases Example

4. Development Phase

Activity:

Unit Testing such as login, search, borrow, return, and admin features.

Testing:

Unit Testing: Test methods like issueBook(), returnBook(), calculateDueDate() individually.

5. Integration Phase

Activity:

Combine all parts of the system (e.g., student login → borrow → update inventory).

Testing:

Integration Testing: Make sure data updates correctly after borrow/return and modules work together.

Testing Phases Example

6. System Phase

Activity:

Now the complete system is running.

Testing:

System Testing: Check the entire process: login → search book → borrow → logout.

7. Acceptance Phase

Activity:

Allow a group of librarians and students to test the system.

Testing:

User Acceptance Testing (UAT): Make sure the system works the way real users expect it to check ease of use, errors, and performance.

Testing Phases Example

8. Release Phase

Activity:

System is installed in the library's live environment.

Testing:

Deployment Testing: Ensure it works with real library data and real user logins.

9. Maintenance Phase

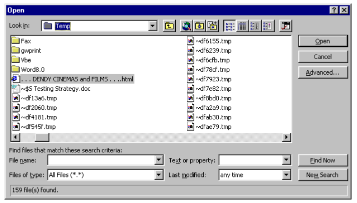
Activity:

After going live, the school updates the library rules (e.g., borrowing period changes from 14 to 21 days).

Testing:

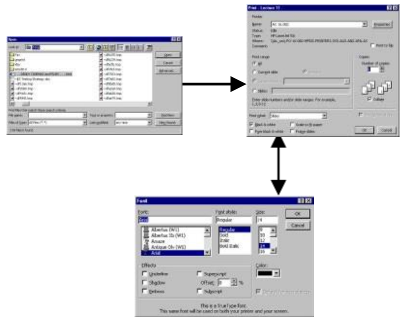
Regression Testing: Test the updated code and retest borrowing/returning to ensure no new bugs were introduced.

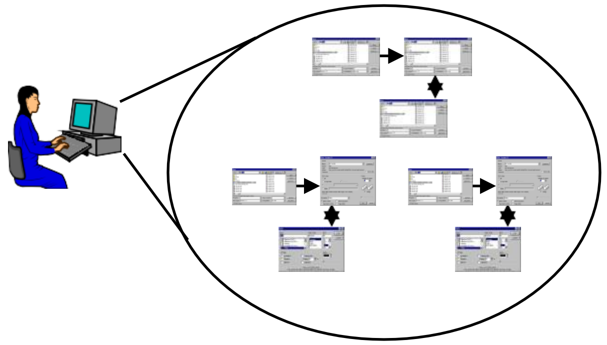
Testing during development



Fax
gwprint
Vbe
Word8.0
. . . DENDY CINEMAS and FILMS . . . .html
~$S Testing Strategy.doc
~df13a6.tmp
~df2060.tmp
~df4181.tmp
~df545f.tmp
~df6155.tmp
~df6239.tmp
~df6cfb.tmp
~df78cf.tmp
~df7923.tmp
~df7e82.tmp
~df8bd0.tmp
~dfa2a9.tmp
~dfab30.tmp
~dfae79.tmp

Find files that match these search criteria:
File name:
Text or property:
Files of type: All Files (*.*)
Last modified: any time
159 file(s) found.





Component Test

- To ensure that each component behaves ‘correctly’.
- Uses white-box testing to check each program function fully.

Integration Test

- To test interaction between related components.
- Focuses on interfaces between components.

System Test

- To ensure that the user requirements have been met.
- Focuses on usual business processes, and normal workflow.

Testing during development

Example Project: Library Management System

The system allows:

- Students to log in
- Search for books
- Borrow and return books
- Admins to add and update book records

1. Component Test (Unit/Module-Level Test)

Test individual functions or methods

What's tested?

- Checks if username and password match
- Returns correct results when a keyword is entered
- Returns the correct return date based on borrowing rules

Ensure each piece of the program behaves as expected in isolation.

Testing during development

2. Integration Test (Connecting Components)

Test how different modules work together

What's tested?

- After a student logs in, can they search and borrow books?
- Does borrowing a book update the inventory and link the record to the student?
- Can the return function correctly update availability and clear dues?

Confirm that combined modules work correctly when used together.

3. System Test (Full Application Workflow)

Test the complete system from the user's point of view

Scenario Example:

A student logs in → searches for a book → borrows it → logs out → logs in later → returns the book

•**What's tested?**

- Login and authentication
- Book search and result display
- Borrowing process (inventory update, due date)
- Returning process (record removal, fine calculation if late)

Ensure the entire system meets user and business requirements in real usage.

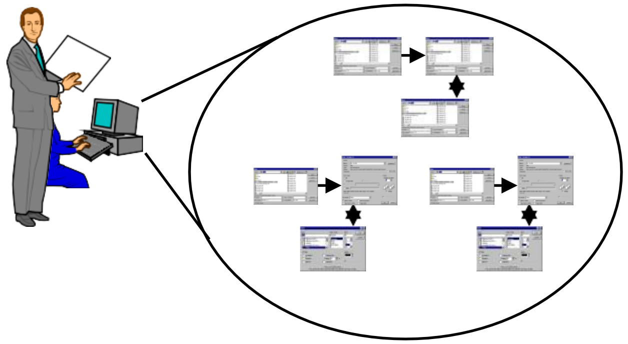
Top-Down vs Bottom-Up

Aspect	Top-Down Integration Testing	Bottom-Up Integration Testing
Testing Direction	Starts from higher-level modules and goes downward	Starts from lower-level modules and goes upward
Initial Components	Real higher-level components; stubs or drivers for lower-level components	Real lower-level components; stubs or drivers for higher-level components
Dependencies	May require stubs (placeholders for lower modules).	May require stubs(placeholders for higher modules)
Advantages	Early validation of system functionality; supports high-level design	Early detection of critical component issues; supports parallel development
Drawbacks	Dependencies on incomplete lower-level components; delayed system testing	May require extensive use of stubs and drivers; complex coordination

Functional vs Non-Functional Requirements in Software Testing

Aspect	Functional Requirements	Non-Functional Requirements
What it describes	What the system should do, its functions and features	How the system should perform, its quality and performance characteristics
Testing focus	Functional Testing: test system behaviors like input validation, workflows, and actions	Non-Functional Testing: test system qualities like speed, reliability, security, and usability
Examples	Login validation, adding to cart, submitting a form	Load time, data encryption, responsiveness under heavy load
Goal	Verify that all features work as intended	Ensure the system meets performance and quality expectations
Based on	User and business use cases	System quality standards and constraints
Test case design	Input-output based tests that simulate real user interactions	Scenario-based tests that evaluate performance, security, or reliability
Who tests	Typically QA testers using black-box or white-box techniques	Often QA + performance/security engineers using tools and benchmarks
Outcome of testing	Confirm correct behavior of individual features	Confirm that the system performs well under expected (and unexpected) conditions

Testing during deployment



Performance Test

- To test system performance under maximum expected load.
- Simulates key processes under maximum load.

Soak Test and Stress Test

- To ensure that system is stable over extended period.
- Load increased until system fails. Checks effects of over-load

Acceptance Test

- Compares system functionality against agreed-on user requirements
- Carried out by client using scenarios, supervised by developer

Example -1

Question 1: Match the Testing Type to the Scenario

- 1. Running a test suite to check if a specific function within a module works as expected.
- 2. Verifying that different modules in the software application can communicate and interact seamlessly.
- 3. Testing the entire software application to ensure that all functional and non-functional requirements are met.
- 4. Evaluating how quickly the software responds to user actions under different levels of load.
- 5. Subjecting the software to extreme usage conditions to identify its breaking points and limitations.
- 6. Having end-users validate whether the software meets their needs and performs as they expect.

Example -1

Question 1: Match the Testing Type to the Scenario

- 1. Running a test suite to check if a specific function within a module works as expected. - Component Testing
- 2. Verifying that different modules in the software application can communicate and interact seamlessly. - Integration Testing
- 3. Testing the entire software application to ensure that all functional and non-functional requirements are met. - System Testing
- 4. Evaluating how quickly the software responds to user actions under different levels of load. - Performance Testing
- 5. Subjecting the software to extreme usage conditions to identify its breaking points and limitations. - Stress Testing
- 6. Having end-users validate whether the software meets their needs and performs as they expect. - User Acceptance Testing

Example -2

Question 2: Match the Scenario to the Testing Type

- 1. Checking if the login button on a website redirects users to the correct dashboard.
- 2. Testing how well different components interact when a user submits a form and the data is processed.
- 3. Verifying that the software works correctly on different browsers and devices.
- 4. Measuring the response time of a web application when 1000 concurrent users access it.
- 5. Stressing a financial software with an exceptionally high volume of transactions to see when it fails.
- 6. Asking real users to try out a new mobile app and provide feedback on their experience.

Example -2

Question 2: Match the Scenario to the Testing Type

- 1. Checking if the login button on a website redirects users to the correct dashboard. - Component Testing
- 2. Testing how well different components interact when a user submits a form and the data is processed. - Integration Testing
- 3. Verifying that the software works correctly on different browsers and devices. - System Testing
- 4. Measuring the response time of a web application when 1000 concurrent users access it. - Performance Testing
- 5. Stressing a financial software with an exceptionally high volume of transactions to see when it fails. - Stress Testing
- 6. Asking real users to try out a new mobile app and provide feedback on their experience. - User Acceptance Testing

Example -3

Question 3: Match the Testing Type to the Scenario

- 1. Verifying that each individual class in the codebase functions correctly and as intended.
- 2. Testing how well the payment gateway module interacts with the customer account module.
- 3. Ensuring that the software's search functionality, login process, and data retrieval work seamlessly together.
- 4. Analyzing how quickly an e-commerce platform loads pages and processes transactions during a flash sale.
- 5. Continuously subjecting the system to a sustained workload over an extended period to assess its stability and reliability.
- 6. Inviting a group of representative customers to test a new software release and provide feedback.

Example -3

Question 3: Match the Testing Type to the Scenario

- 1. Verifying that each individual class in the codebase functions correctly and as intended. - Component Testing
- 2. Testing how well the payment gateway module interacts with the customer account module. - Integration Testing
- 3. Ensuring that the software's search functionality, login process, and data retrieval work seamlessly together. - System Testing
- 4. Analyzing how quickly an e-commerce platform loads pages and processes transactions during a flash sale. - Performance Testing
- 5. Continuously subjecting the system to a sustained workload over an extended period to assess its stability and reliability.- Soak Testing
- 6. Inviting a group of representative customers to test a new software release and provide feedback. - User Acceptance Testing

Performance test

Modelling the expected usage of a software program by simulating multiple users accessing the program concurrently.

Important for multi-user systems, often using client/server model, such as web servers.

Examples

Simulate 500 students logging in and searching for books at the same time to check if the system handles the maximum expected load without slowing down.

Soak Testing

Testing with a significant load extended over a significant period of time, to discover how the system behaves under sustained use

Examples

Keep the system running with constant activity (borrowing/returning books) over 48 hours to check if there are memory leaks or slowdowns over time.

Stress Testing

Subjecting a system to unreasonable load, while denying it the resources needed to process that load, (RAM, disc, mips, interrupts, etc).

Examples:

Force the system to handle 10,000 book searches or logins at once, which is beyond normal usage, to see if it crashes or fails gracefully.

Load VS Stress Vs Soak

Test Type	Purpose	Test Scenario	Example Question
Load Test	Assess performance under expected load conditions	Simulate concurrent users or transactions within expected limits	Can the system handle 1000 users concurrently?
Soak Test	Evaluate system stability over time	Sustain load over an extended period (hours/days)	How does the system perform after 72 hours of continuous usage?
Stress Test	Identify breaking points and recovery behavior	Push system beyond its limits, observe failures and recovery	What happens when the system receives 10 times the normal traffic?

Why is it hard?

– Ariane 4 rocket

European Space Agency (ESA).

– Ariane 5 rocket

Same software. So, what went wrong? The 64-bit Inertial Reference System was forced into a 16-bit system, causing an overflow error.

Software that works in one environment does not necessarily work in another. Proper testing and validation would have prevented this disaster.

Nectar Card Fiasco : Tuesday, 17 September 2002

http://news.bbc.co.uk/1/hi/business/2268797.stm

Another case study: **Nectar Card Fiasco,** a loyalty card scheme that failed **spectacularly** due to poor testing.

- **Retail giants** like Sainsbury's, BP, and Barclays launched an online reward system.
- **100 points** were awarded for signing up, and **500 points could get you a Big Mac!**
- Millions of people rushed to sign up, but the **website crashed repeatedly.**
- Users kept trying again and again, **increasing the traffic,** making things worse.

Performance testing should **anticipate high traffic spikes** before a major campaign, especially for **consumer-facing applications.**

The system collapsed!

As the **deadline** approached, the website was hit with **10,000 requests per hour.**

- **Massive advertising and TV promotions** brought even more traffic.
- The website was **not stress-tested**, and it **collapsed** under the pressure.
- IT and marketing teams **did not communicate effectively,** leading to a **failure of risk assessment.**
- *"The online operation was simply taken by surprise by the demand".* Ian Barber, Barclaycard

  "All I can think of here is that marketing has not been properly communicating with IT. To send this volume of letters out driving people to the website and not have the capacity in place is a serious flaw".
  Andrew Didcott, Internet expert)

What do you think went wrong?

- Was it a problem of testing?
  - The functionality of the system had been tested, but not under load
  - Performance testing had not anticipated the level of user load
  - Nobody knew that the system would fail completely under pressure

- Was it *really* a problem of testing?
  - Why wasn't it tested at the right level?
    - Should it have been tested with 10,000 hits/hour?
    - But what if it then had 50,000 hits/hour?   500,000 hits/hour?
  - Test management….

Another testing story

- A global grocery and general merchandising retailer
  - Grocery market leader in the UK
  - 702 stores
  - 30% market share
  - Stores in 14 countries
- In 2002 new online shopping system was to be rolled out
- Management demanded 'no glitches'

The test plan

- Total budget for testing, £1 million
- Testing to be carried out off line (so as not to interfere with live system)
  - **Capacity model:** Simulated user load two years into the future.
  - **Usage model:** Simulated a typical mix of tasks, ensuring the platform handled various user behaviors.
  - **Test database:** Used a full-sized database to ensure that performance issues related to large datasets were accounted for.
  - To include **soak** and **stress** testing

Test Cases

Test Cases

A **test case** is a **step-by-step set of instructions** used to check whether a specific part of a software system is working as expected.

It describes:

- **What to test**
- **How to test it**
- **What result is expected**

Basic Components of a Test Case

Element	**Description**
Test Case ID	A unique name or number (e.g., TC001)
Test Scenario	What feature or function you're testing (e.g., "Login with valid credentials")
Test Steps	Step-by-step actions to perform the test
Expected Result	What the system should do if it's working correctly
Actual Result (optional)	What actually happened during testing
Pass/Fail	Whether the test matched the expected result

Simple Example (Library System)

Field	Value
Test Case ID	TC001
Test Scenario	Login with correct Student ID and password
Test Steps	1. Go to login page 2. Enter valid ID and password 3. Click login
Expected Result	User is logged in and redirected to the dashboard
Actual Result	User logged in successfully
Pass/Fail	Pass

Test Case 2: Invalid Login Attempt

Field	**Value**
Test Case ID	TC002
Test Scenario	Login with incorrect Student ID or password
Test Steps	1. Go to login page 2. Enter incorrect ID or password 3. Click login
Expected Result	System shows error message: “Invalid ID or password”
Actual Result	System displayed error as expected
Pass/Fail	Pass

Test Case 3: Search for a Book by Title

Field	**Value**
Test Case ID	TC003
Test Scenario	Search for a book using the title
Test Steps	1. Login 2. Go to Search 3. Enter book title 4. Click search
Expected Result	Matching books are displayed with status
Actual Result	List of matching books shown
Pass/Fail	Pass

Test Case 4: Borrow Available Book

Field	**Value**
Test Case ID	TC004
Test Scenario	Borrow a book with available copies
Test Steps	1. Login 2. Search for book 3. Click “Borrow”
Expected Result	Book is borrowed and due date is shown
Actual Result	Book borrowed successfully
Pass/Fail	Pass

Test Case 5: Borrow Unavailable Book

Field	**Value**
Test Case ID	TC005
Test Scenario	Attempt to borrow a book with 0 available copies
Test Steps	1. Login 2. Search book 3. Try to borrow
Expected Result	System displays “No copies available”
Actual Result	Error message shown as expected
Pass/Fail	Pass

Test data and test scenarios

Estimating target load: Extrapolating historical data

Example: Online Bookstore Order Forecasting

Scenario: An online bookstore is growing steadily. The company wants to estimate how many orders it will receive during the next holiday season to make sure its website and delivery system can handle the load.

Historical Data (Observed):

From July to September, the company recorded:

- July: 10,000 orders
- August: 12,000 orders
- September: 14,000 orders

This shows a clear upward trend of +2,000 orders per month.

Estimating target load: Extrapolating historical data

Extrapolation:

If the same trend continues, they can predict future orders:

- October: 16,000 orders
- November: 18,000 orders
- December: 20,000 orders

This is extrapolating the trend based on past data to estimate future demand.

Use of Extrapolated Data:

Using these predictions, the company can:

- Make sure its servers can handle up to 20,000 orders in December.
- Prepare enough staff and delivery resources.
- Run performance tests on the website to handle peak traffic.

Another example

– Australian census 2016

AUSTRALIA
PEOPLE
23,401,892
50.7%
Female
49.3%
Male
HOUSEHOLD INCOME
Average weekly (gross)
$1,234
$1,438
2011
2016
AGE
15-64
0-14
65+
18.7%
65.6%
15.7%
AVERAGE PEOPLE PER HOUSEHOLD
2.6
TOP 5 COUNTRIES OF BIRTH
3.9%
England
2.2%
China
1.9%
India
66.7%
Australia
2.2%
New Zealand
DWELLINGS
Owned outright
31.0%
Mortgage
34.5%
Rented
30.9%

– What went wrong?

"On 9 August 2016, at about 7.30pm, the census website was hit by distributed denial-of-service attacks. The site was flooded with traffic in an attempt to overload it and shut it down. Along with a hardware failure, and a false report that data was at risk, there was widespread concern about the security of the census. Just after 8pm the ABS took the website offline for 40 hours.

In his memoir, A Bigger Picture, the former prime minister Malcolm Turnbull described the event as a "humiliating debacle for a government that was promoting innovation, agility and the promise of the digital era".

Initially, the ABS and IT contractor IBM stated there had been a "massive cyberhack" and implied it had been the work of a foreign state. But as Turnbull noted, the DDoS attacks were "quite modest in scale" and a result of a failure on IBM's part to deliver on its contractual obligations around DDoS protection.

The Guardian Newspaper, 11/7/2021
https://www.theguardian.com/australia-news/2021/jul/11/censusfail-hangs-over-australian-bureau-of-statistics-as-it-prepares-for-2021-survey

Pos

- The Tesco online shopping system performed without a hitch since it went live in May 2002

- In terms of revenue, Tesco is one of the 10 biggest retailers in the world.

Reminders

- Check Ed regularly!

- Attend tutorials – and engage!
  - (In general, this has been fantastic this semester)
  - Updated approach to participation assessment

The End