Why children should learn to test - why anyone should test?

Kari Kakkonen

JFTL conference

September 14th, 2021

Kari Kakkonen

twitter.com/kkakkonen
linkedin.com/in/karikakkonen/
Dragonsout.com

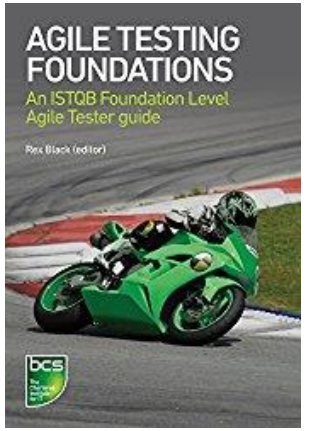
- **ROLES**
  - Knowit Solutions Oy, Director of Training and Competences, Lead Consultant, Trainer and Coach
  - Children's and testing author at Dragons Out Oy
  - Treasurer of Finnish Software Testing Board (FiSTB)
  - TMMi Board of Directors
- **ACHIEVEMENTS**
  - ISTQB Executive Committee 2015-2021
  - Influencing testing since 1996
  - Ranked in 100 most influential IT persons in Finland (Tivi magazine)
  - Great number of presentations in Finnish and international conferences
  - TestausOSY/FAST founding member.
  - Co-author of Agile Testing Foundations book
  - Regular blogger in Tivi-magazine
- **EDUCATION**
  - ISTQB Expert Level Test Management & Advanced Full & Agile Tester certified
  - DASA DevOps, Scrum Master and SAFe certified
  - SPICE provisionary assessor certified
  - M.Sc.(Eng), Helsinki University of Technology (present Aalto University), Otaniemi, Espoo
  - Marketing studies, University of Wisconsin-Madison, the USA.

BUSINESS DOMAINS

- Wide spread of business domain knowledge
  - Embedded, Industry, Public,
  - Training, Telecom, Commerce,
  - Insurance, Banking, Pension

SERVICES

- ISTQB Advanced, Foundation and Agile Testing
- A4Q AI and Software Testing
- Knowit Quality Professional
- DASA DevOps
- Quality & Test process and organization development, Metrics
- Agile testing, Scrum, Kanban, Lean
- Leadership
- Test automation, Mobile, Cloud, DevOps, AI
- Quality, Cost, Benefits.



Creating a Nordic powerhouse for digital solutions with a sustainable impact

knowit ♡ CYBERCOM GROUP

3,800+
/ Experts

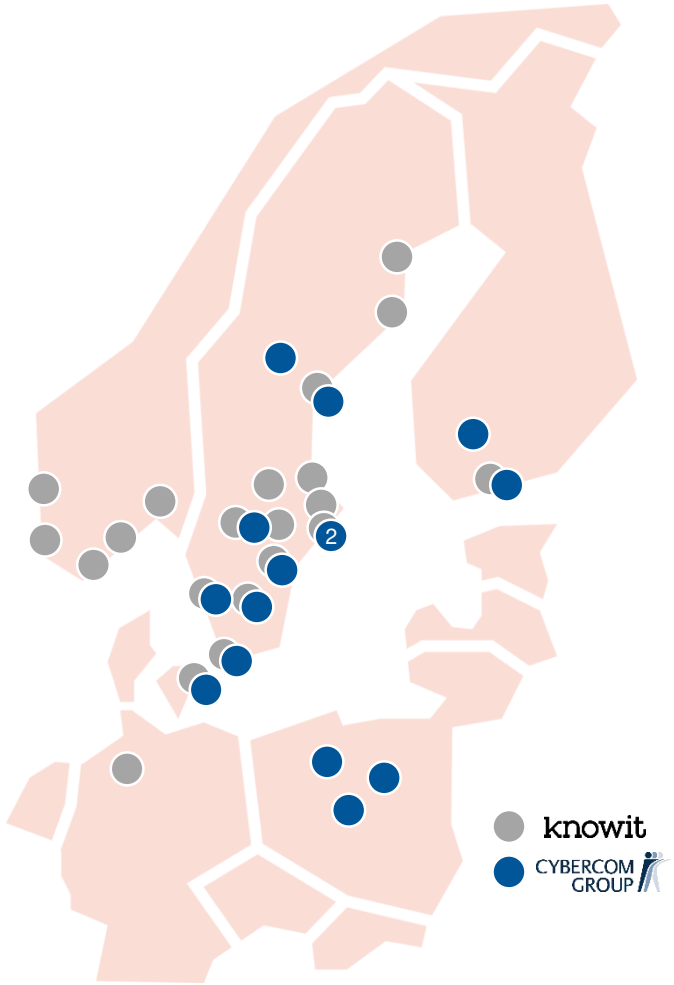
4 Business Areas
/ Solutions, Experience, Insight ja Connectivity

545 M €
/ Liikevaihto

6 Countries
/ Sweden, Norway, Finland, Denmark, Germany and Poland

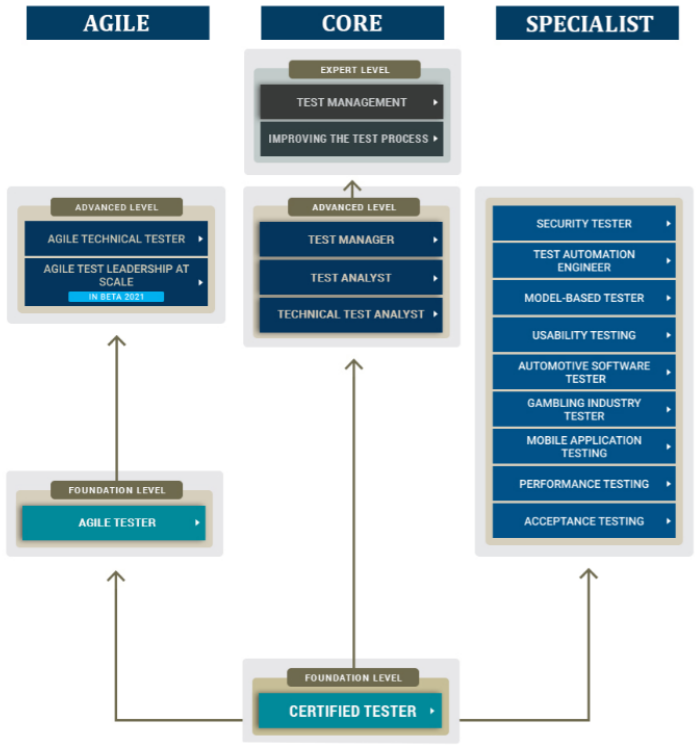
Nordic ESG champions
/ Clear vision to accelerate the UN Environment, Social, Governance, and Sustainable Development agenda

54,3 M €
/ EBITA



ISTQB GLOBAL PRESENCE

- Number of exams administered: **1 030,000**
- Number of certifications issued: **750,000**
- **In 129 countries**

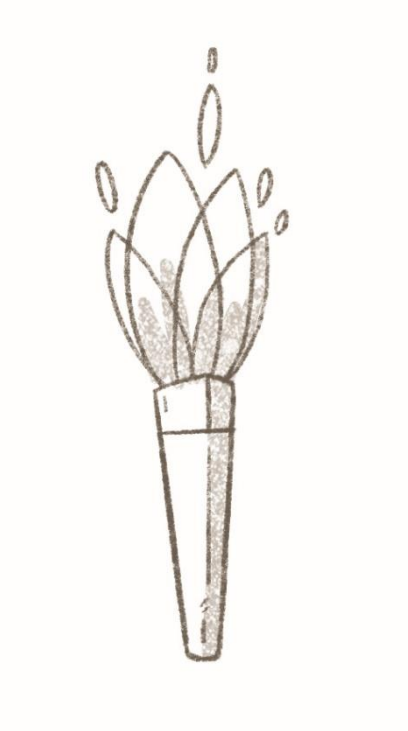


Agenda

- Book project recap
- Fantasy as a way to learn software testing
- Feedback survey results from book readers
- Interesting ways of learning for children
- Interesting ways of learning for adults
- Exercise: Draw your own knight
- Q&A

The book project recap

I want to tell a story



“Every person has a story. Every cause needs a storyteller. Learn to be a storyteller because unless you are a candidate for a reality show, no one else is going to tell your story for you.

So tell us a story. Tell us a good story. And let that good story be one part of a symphony of stories that makes this world a better place.”

James Whittaker

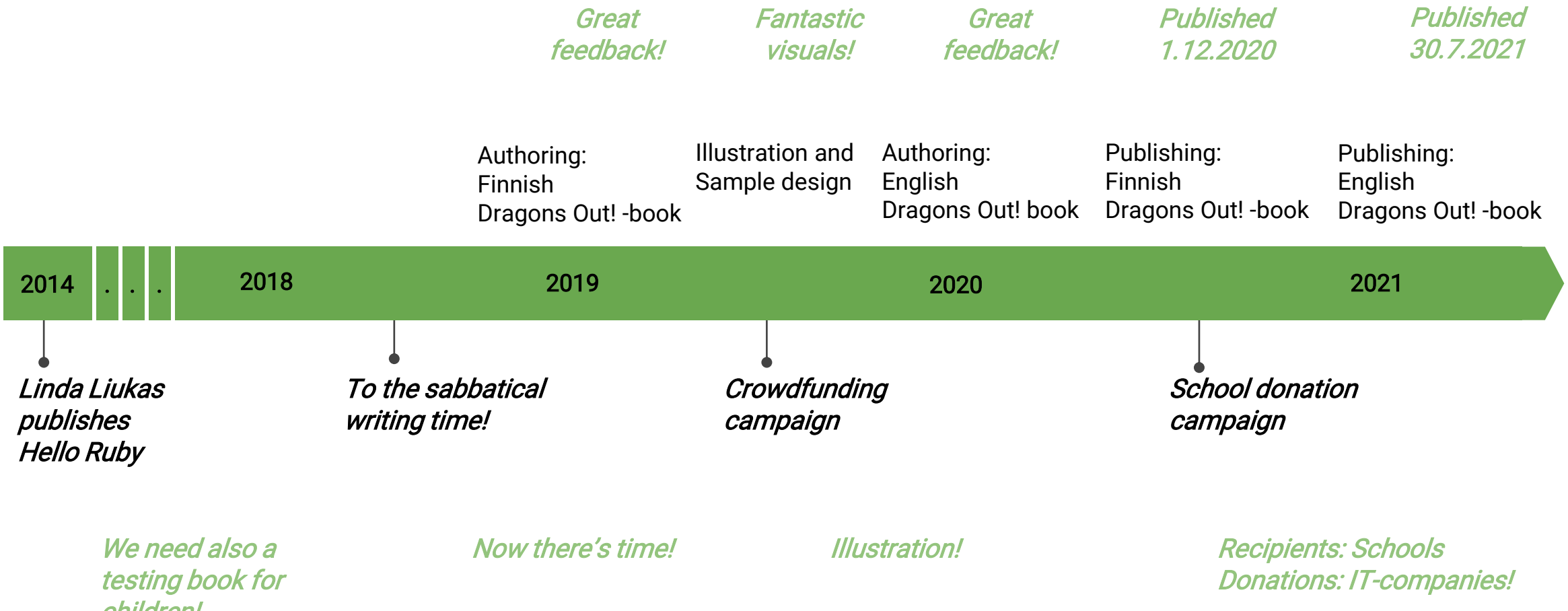
https://medium.com/@docjamesw/the-storytelling-manifesto-f17548a358b3

Why testing for children?

- Coding has started to interest children and youth.
  - Many parties in our societies promote coding e.g. via coding schools
- There is not enough software testing education
  - Testing is even over half of all software development work
  - **There is also a lack of testers, not only a lack of coders**
  - Testing is the new basic skill
- Good quality is needed in software development in Finland and around the world
- We need to make software testing familiar already to children
  - Testing schools
  - Testing books

- My own solution, on top of all the coding schools, is to offer children a book about software testing

Book project highlights



About the book ”Dragons Out!”

- Author Kari Kakkonen
- Illustrator Adrienn Széll
- Text and illustration rights Dragons Out Oy
- A version of this presentation is available for teachers (or anyone) under Creative Commons –license at the book web site
  - Translated to many languages!
- More info: www.dragonsout.com

Fantasy as a way to learn software testing

Power of the story

Story

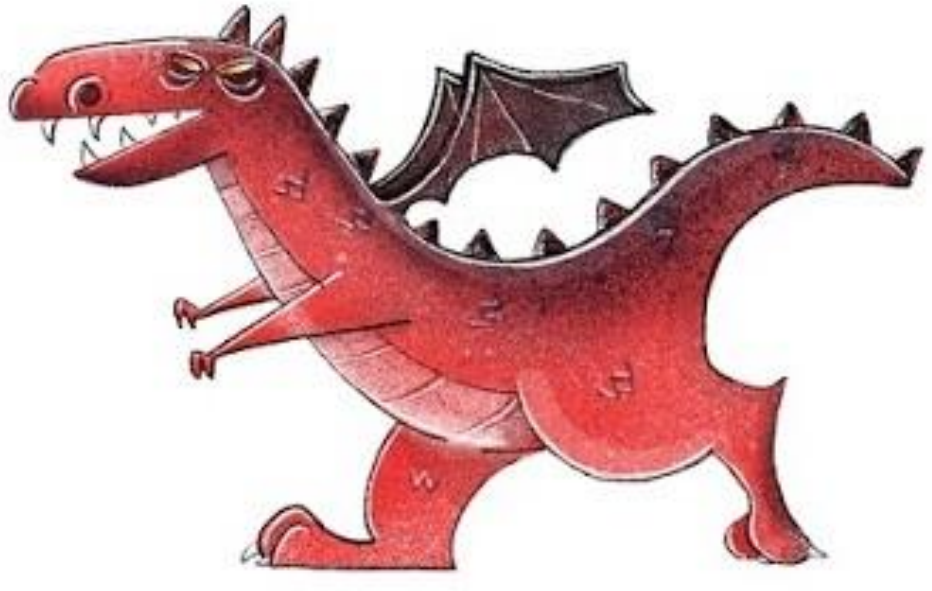
- Swanlake turned her horse around and rode fast back to the palisade. She called to the knights and building master Aidan that the dragon was coming. All the sharpened trunks needed to be moved urgently to the hole in the palisade. Spears and swords, whoever had them, should be fetched immediately. All available water should be poured into buckets. Then she went to find Yellowbeard at the castle.

Explanation

- *In the story the dragon arrives to a village in the middle of the repairs of the palisade. Similarly, most of the defects are found in software during software development, before the software is released. Then the* ***people who look for defects (testers) and fix defects (coders),*** *are always available. Usually a tester finds the defect, so doesn't wait for a user to find the defect later. In this story Swanlake was a tester who found and identified the defect, that is the dragon. As a tester she couldn't this time fix the defect but needed coders (developers) to help.*

Annoying dragon

- **Color:** Red
- **Size:** Medium
- **How difficult to find?:** Difficult
- **How difficult to get rid of?:** Easy
- **Flies?:** No
- **Wings:** Small
- **Breaths fire?:** Yes
- **Favorite thing:** Eating lambs

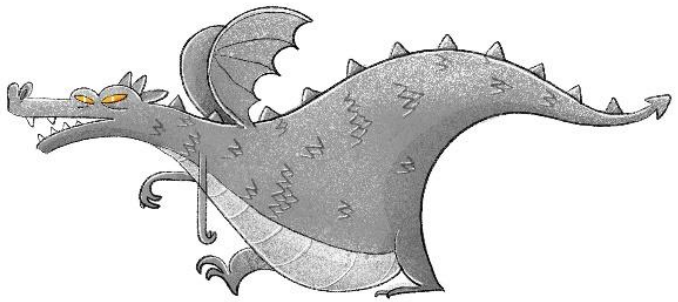


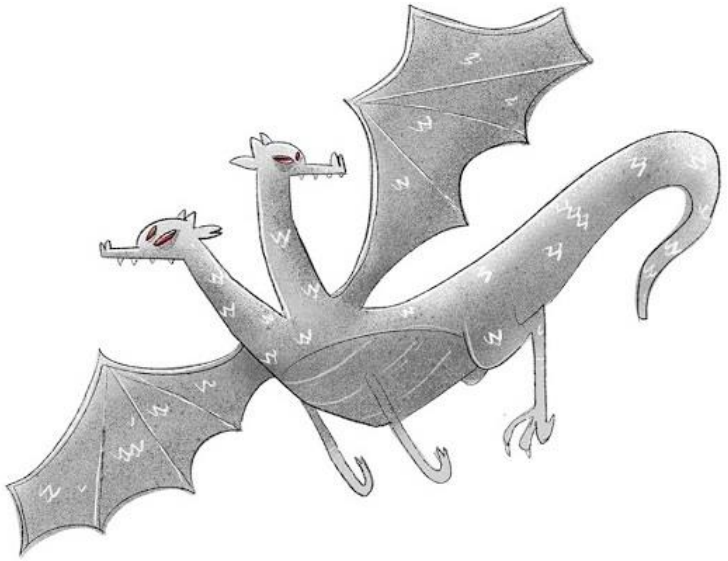
Annoying dragon

- **Defect name:** Memory leak
- **Severity:** Medium
- **Defect symptoms:** The computer gets slower, until it can’t function at all, and it shuts down
- **Cause of the defect:** Memory is reserved for use of the software, but it is not freed after usage
- **Root causes:** Developer is not careful in freeing the memory. May not know how, may not remember.
- **Testing:** You measure used memory as you use the software. If the amount of used memory increases all the time, it is probably a memory leak.
- **Fix:** You run the software one line of code at a time, until you find the spot that should be fixed. Memory is released with a proper piece of code.

Robbing dragons

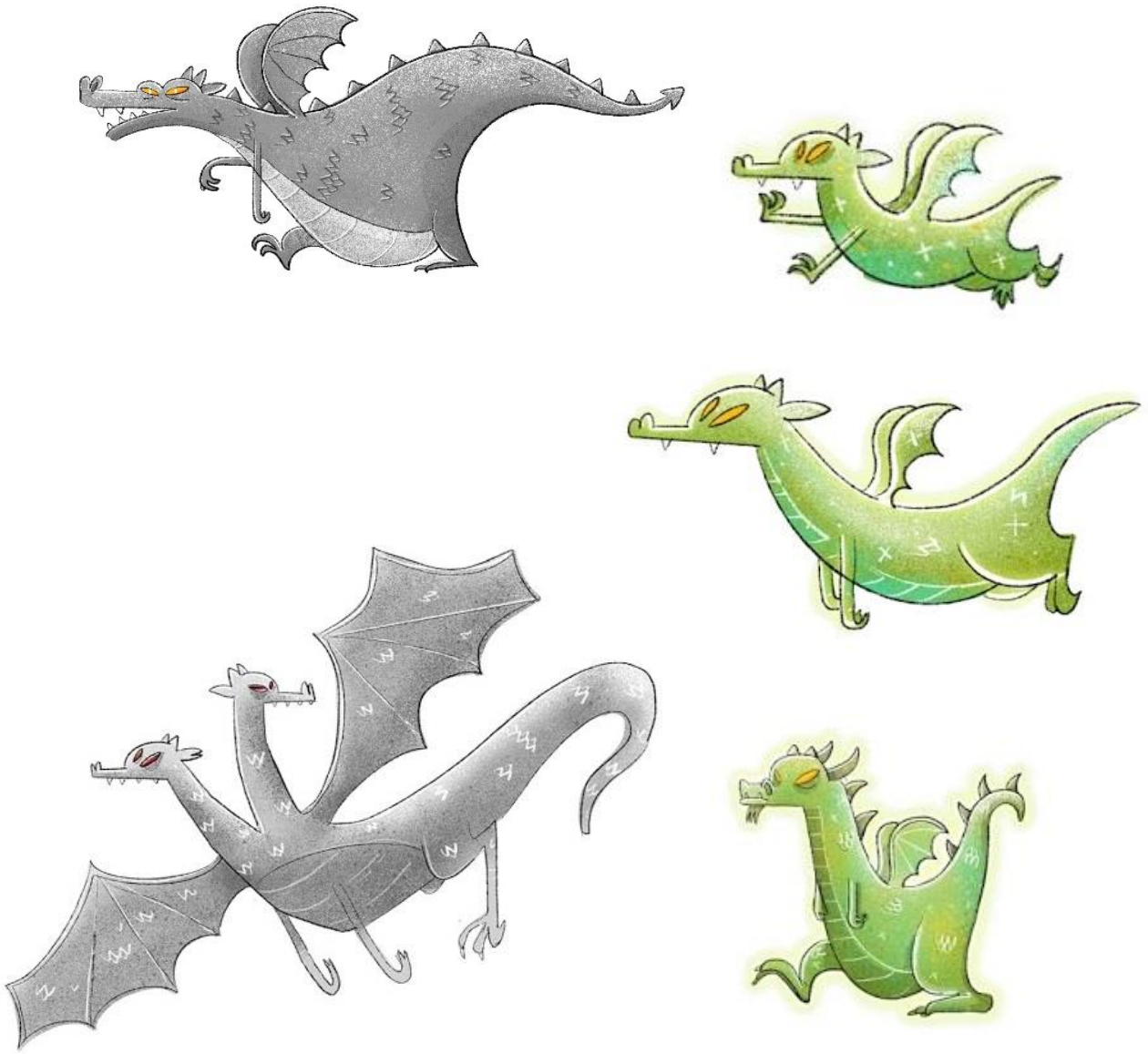
- **Color:** Glittering green, or gray
- **Size:** Small to large
- **How difficult to find?:** Easy to difficult
- **How difficult to get rid of?:** Easy to difficult
- **Flies?:** Some fly, some don’t
- **Wings:** Small to large
- **Breaths fire?:** Yes
- **Favorite thing:** Stealing food and treasure




Robbing dragons

- **Defect name:** Functionality defects
- **Severity:** Low-Medium-High
- **Defect symptoms:** The software doesn't do what it should do. Calculation gives wrong result. User sees information in the wrong place.
- **Cause of the defect:** The functionality has been coded wrong.
- **Root causes:** Developer has not understood, what the user has meant. Or the defect exists due to carelessness, or hurry.
- **Testing:** You use the software normally, based on tester experience or requirement definitions.
- **Fix:** Code is changed to work correctly.



Mean dragon

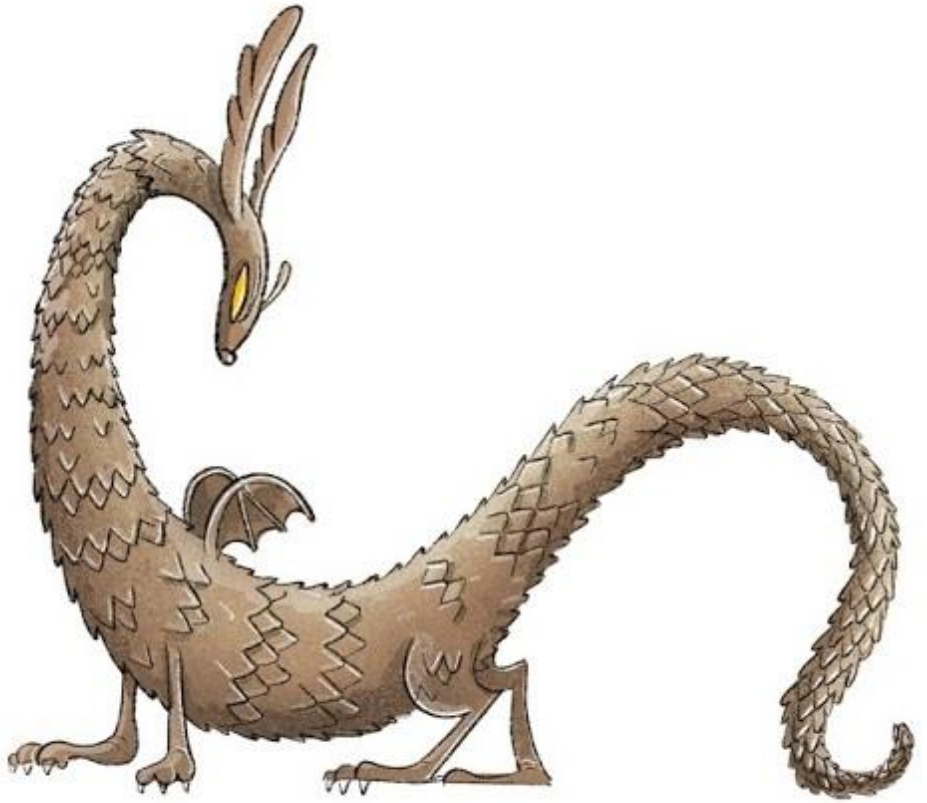
- **Color:** Black
- **Size:** Small
- **How difficult to find?:** Difficult
- **How difficult to get rid of?:** Medium
- **Flies?:** Yes
- **Wings:** Medium
- **Breaths fire?:** A lot
- **Favorite thing:** Stealing food and treasure without being detected

Mean dragon

- **Defect name:** Security defect
- **Severity:** High
- **Defect symptoms:** Information from the software is found outside the system (e.g. bank card information). It could also be just software functioning wrong.
- **Cause of the defect:** A criminal has used security defect to break into the system, and then has stolen or destroyed something.
- **Root causes:** Developer has not followed the latest secure coding principles. Maybe doesn't know these.
- **Testing:** You look for known vulnerabilities in the software by using it, or via a security testing software. You can also review code. Checklist of known defects helps.
- **Fix:** A known vulnerability has also a known fix. It is fixed in the code or system settings.

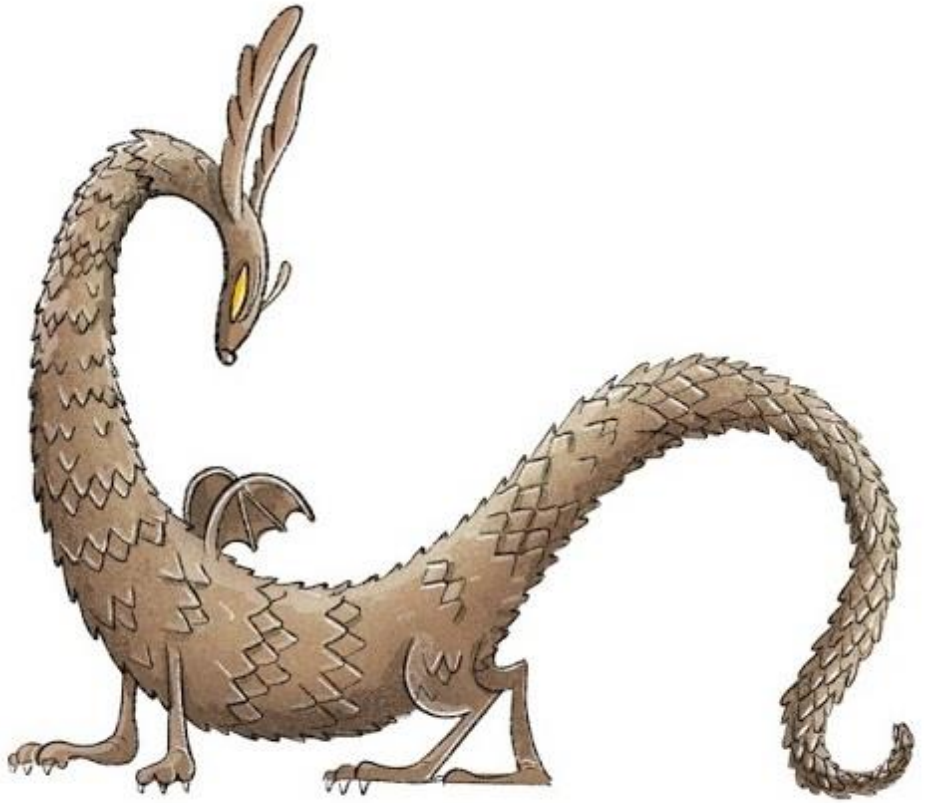
Underground dragon

- **Color:** Brown
- **Size:** Large
- **How difficult to find?:** Easy
- **How difficult to get rid of?:** Medium
- **Flies?:** No
- **Wings:** Small
- **Breaths fire?:** A lot
- **Favorite thing:** Finding easy food and eating



Underground dragon

- **Defect name:** Hardware defect
- **Severity:** High
- **Defect symptoms:** Some part of or all of the computer doesn’t work.
- **Cause of the defect:** A part of hardware has broken over time.
- **Root causes:** A part of hardware may be of low quality, so it doesn’t last as long as it should. Possibly the part doesn’t work well with other parts, so it breaks.
- **Testing:** You use the system normally. You observe the hardware. Test environment uses similar hardware than the users will have.
- **Fix:** You change a broken part to a new one or change to a part that better fits other parts.



Nice dragon

- **Color:** Glittering green
- **Size:** Medium
- **How difficult to find?:** Easy
- **How difficult to get rid of?:** Easy
- **Flies?:** Yes
- **Wings:** Medium
- **Breaths fire?:** Yes
- **Favorite thing:** Eating animals and helping people

Nice dragon

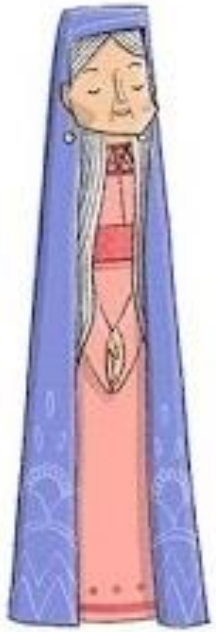
- **Defect name:** Defect seeding, mutation testing - a defect created on purpose
- **Severity:** Low
- **Defect symptoms:** It looks like the functionality works wrong, e.g. wrong result from a calculation. So, the defect looks like a functionality defect.
- **Cause of the defect:** Tester or coder has created the defect into the code on purpose.
- **Root causes:** The idea is that when all seeded defects have been found, all defects have been found.
- **Testing:** You use the system normally and try to find all seeded defects. You will also find real defects. When the last seeded defect is found, you can stop testing.
- **Fix:** Remember to fix the code also for the seeded defects, in the same way as for real functionality defects.

Knights

- Developers
  - Programmers, coders
  - Testers
- Work together, usually in the same development team (Agile)
- Build software
- Test software
- Find and fix defects

Children, villagers

- Users
- Help build software
- Test new software
- Test old software
- Ask for help from technical support and developers, when needed

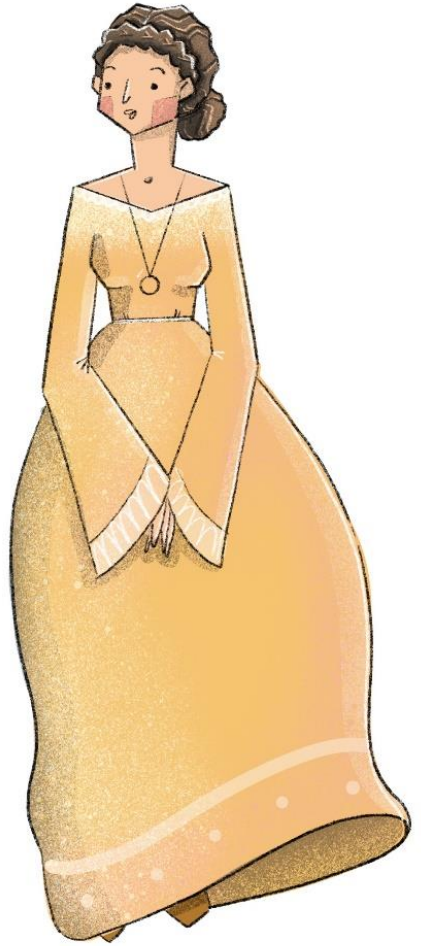


Hunters

- Technical support
- Maintain the software / system
- Test
- Fix defects
- Help users
- Ask for help from developers when needed
- Sometimes in the development team (DevOps team)

Lords and Ladies

- Order software and systems
- Product owners
- Management
- Define what the software should do
- Listen to developers



Sages

- Experts in
  - Usability
  - Security
  - Performance
- Help product owners
- Help development teams

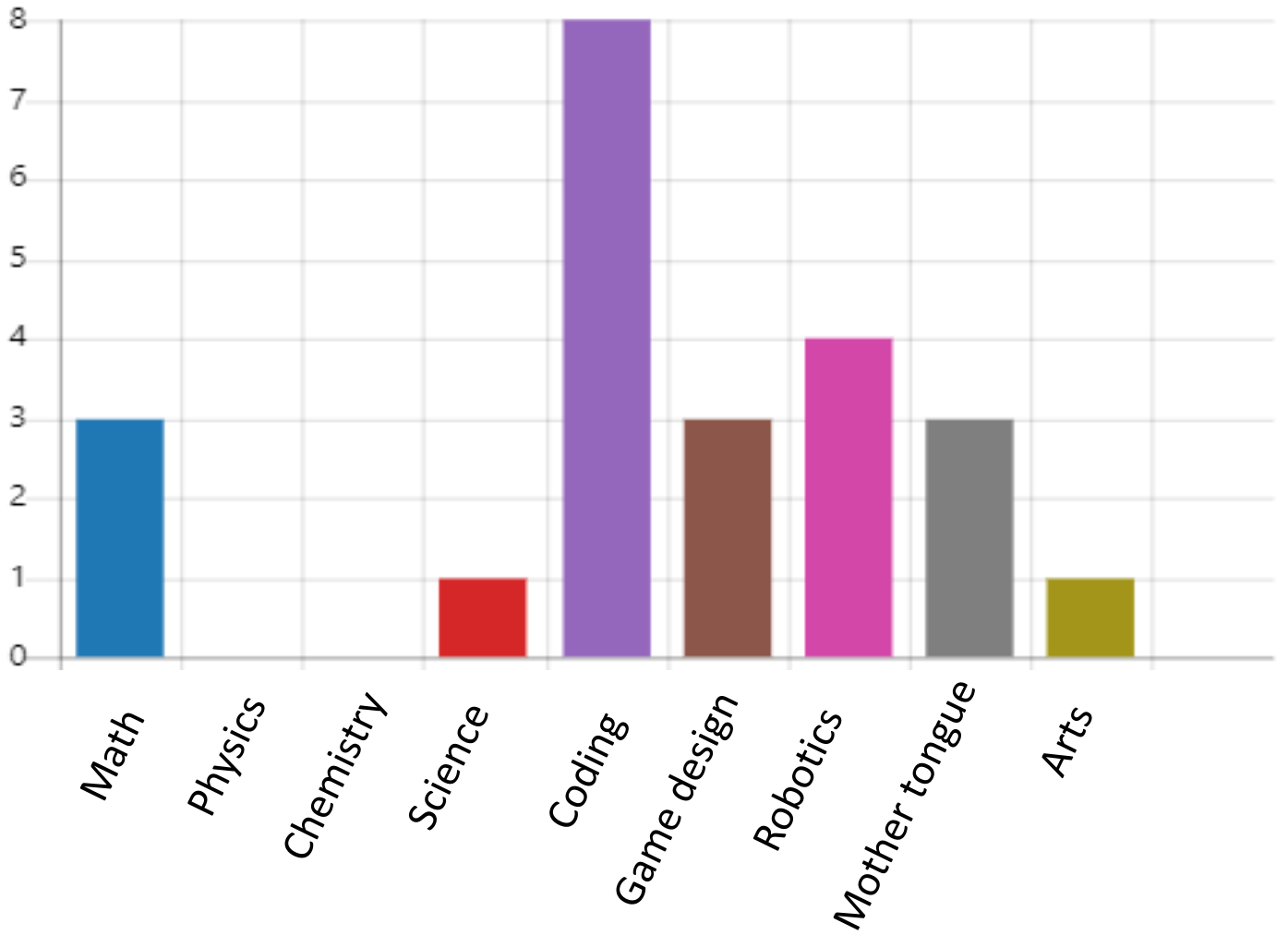
Feedback survey results from book readers

About collecting feedback

- I've run a survey to teachers in Finland about what kind of learning approach has worked best for students of different ages.
- The biggest interest has been in **age groups 10-15**.
- The feedback is great.
  - Survey results combined with verbal feedback from teachers
  - On a scale from 1-5 the book/testing rates 4,22
- Numbers are small and thus should be treated as tentative
  - However, the responses reveal interesting views

Software Testing coupled with fantasy fits to many teaching subjects

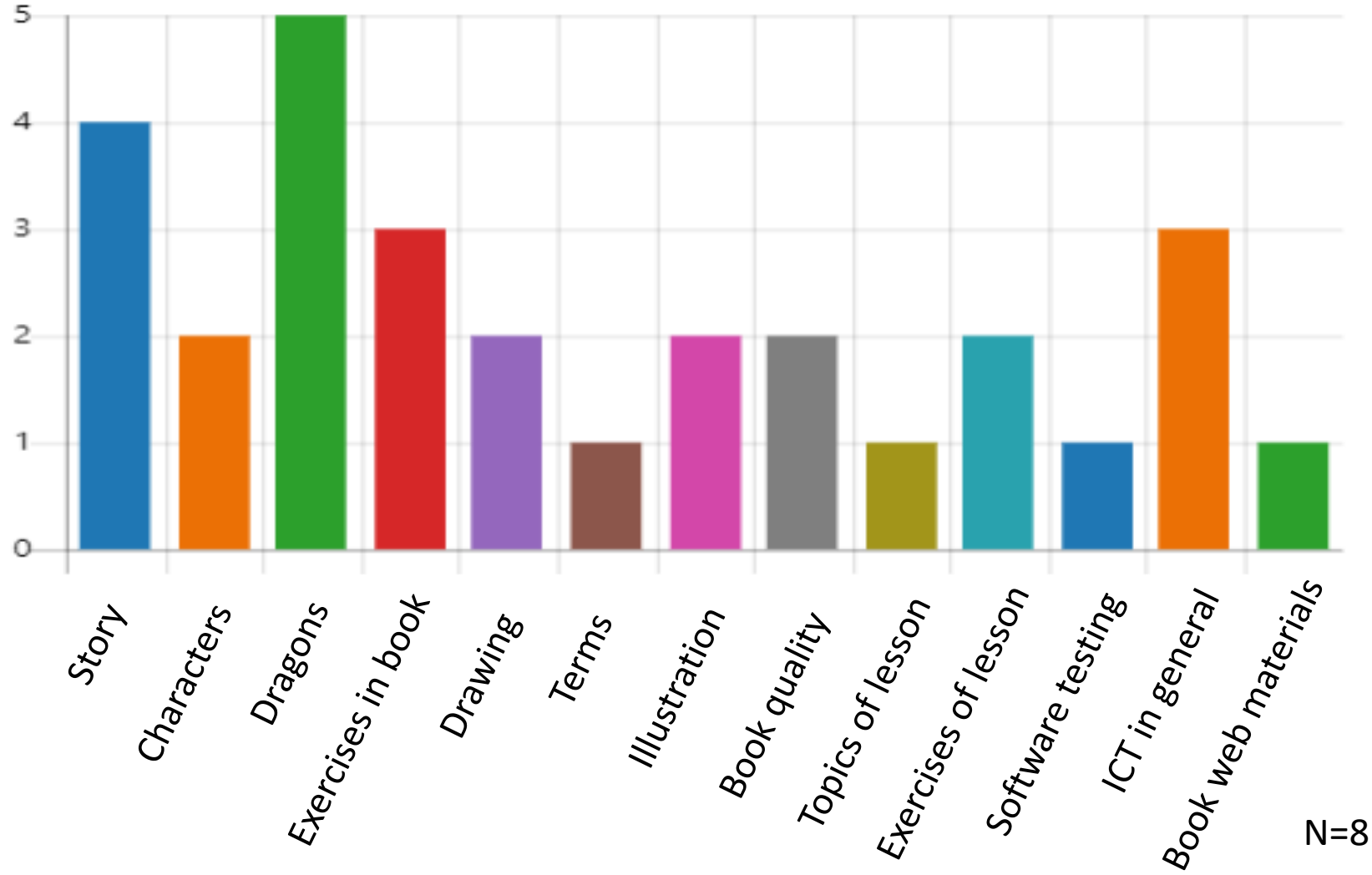
- Multiple teaching subjects are covered
  - Phenomena learning
- Coding is most (not surprisingly) best match with software testing



N=8

What interests children about the topic?

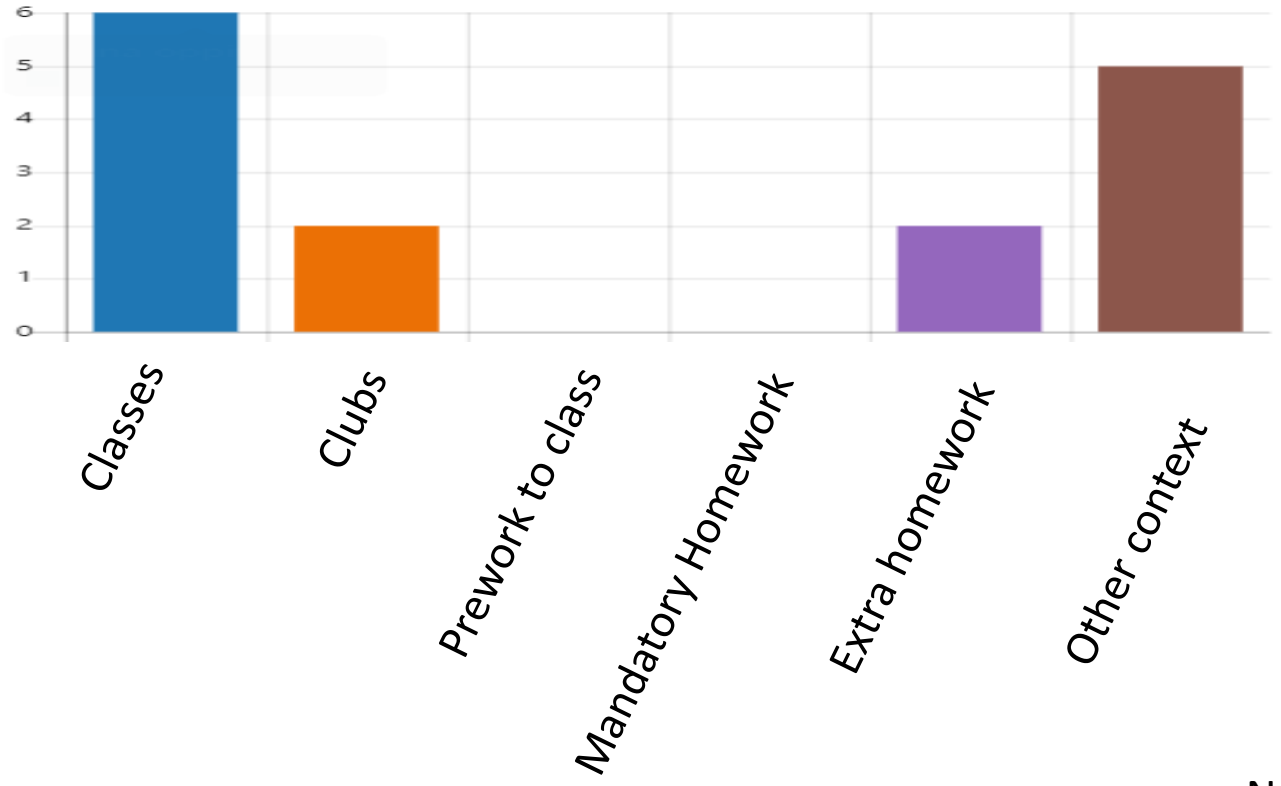
- Fantasy works best in creating attraction
- Learning of software testing comes as a side effect
- A book is preferred over powerpoint lesson (ppt)



N=8

What are best situations to teach with a software testing book?

- Teacher assisted learning (classes) gets highest rating even though books can of course be read individually
- Organized learning is best!



N=8

Combining learning approaches

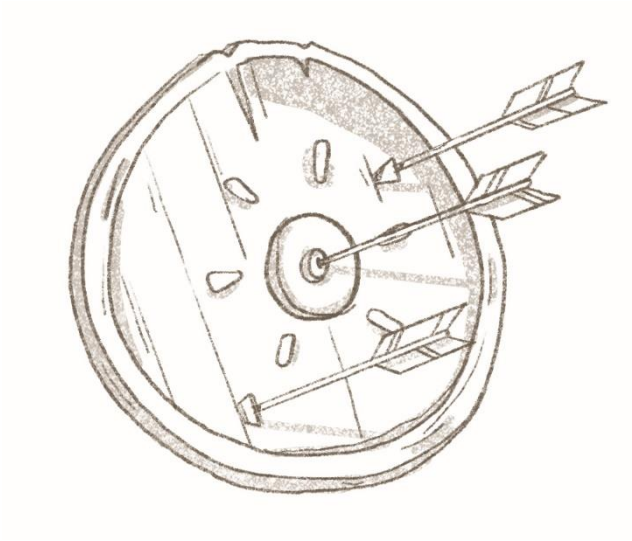
- It has been fascinating to see how **combining different learning approaches** works in getting enthusiastic learners into software testing.
- The usual combination has been
  - **drawing** exercises
  - **listening** or **reading** testing content
  - understanding through the power of **analogies** between fantasy and software testing
  - **exploratory** testing.

Interesting ways of learning for children

How children learn?

- Stories, examples
- Identifying with others, Idols
- Imitating
- Rhymes, songs
- Playing, games
- Exploring, doing, trial and error
- Simplicity, clarity
- Repeating
- Remembering
- Boundaries (right and wrong)

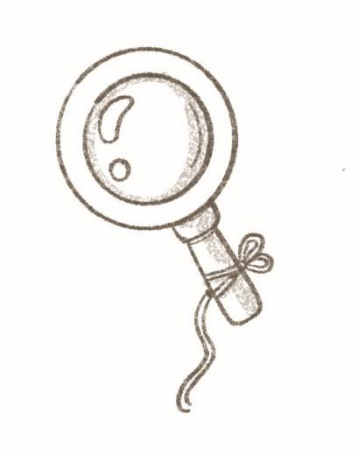
Many learning strategies is a good thing



- “Children and teenagers learn by observing, listening, exploring, experimenting and asking questions” (1)
- “The broader the range of strategies that children can use appropriately, the more successful they can be in problem solving, in reading, in text comprehension and in memorizing. “ (2)

1 https://raisingchildren.net.au/school-age/school-learning/learning-ideas/learning-school-years
2 http://www.ibe.unesco.org/sites/default/files/resources/edu-practices_07_eng.pdf

Storytelling works for all learning styles

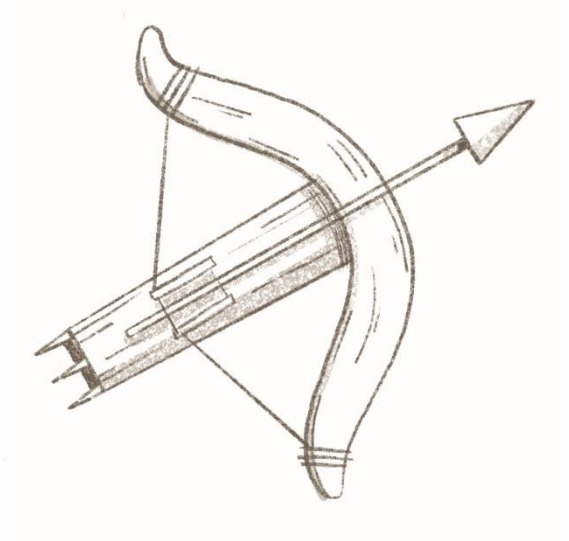


- Visual learners like the mental pictures they get from storytelling
- Auditory learners connect with the words and the storyteller's voice.
- Kinesthetic learners can hook into the emotional connections and feelings from the story.

- Storytelling also helps with learning because stories are easy to remember

https://www.harvardbusiness.org/what-makes-storytelling-so-effective-for-learning

Focus on 10-12 year olds or “Tweens”



- “Around the age of 11 or 12, children learn to think about abstract concepts.”
- “Tweens display strong metacognition skills, i.e. ability to think about thinking. Children display this ability through an awareness of knowledge, an awareness of thinking, and an awareness of thinking strategies.”
- Software testing is essentially about thinking what we already know and expanding that knowledge by exploring.

https://www.scholastic.com/parents/family-life/creativity-and-critical-thinking/development-milestones/cognitive-development-11-13-year-olds.html

6 ways for children to learn testing - takeaways

- Start with a Fantasy example, explain into ICT-world
- Be extremely clear and concise
- 5-minutes of theory, 20-minutes of exercise structure
- Use all the senses (listen, see, talk, draw dragons)
- Use common sharing of exercise results (e.g. Padlet)
- Try out your test ideas immediately to an app of your choice

Interesting ways of learning for adults

6 ways for adults to learn testing - takeaways

- Use examples and analogues from real-life
- Be extremely clear and concise
- Hands-on, mostly exercises in the learning
- Use all the senses (listen, see, talk, draw mindmaps)
- Use common workspace for real-time status of testing (e.g. Mural, Miro)
- Get your hands dirty and test some (buggy) software immediately & explain how you test it

- These are extrapolated from the findings how children learn best.

Analogues, parallells

- Talk about day-to-day life to drive your point
  - Cars
  - Hobbies
  - Sports
  - Pets
  - Family

Pic: https://medium.com/serious-scrum/scrum-s-connection-to-rugby-597405fed5ec

Clear, concise communication

- Express what you mean clearly
- Start from big picture
- Use concepts that sum it all up, e.g.
  - Keywords
  - Mission statements
  - Vision statements
  - Values

Quote: Farhshad Asl
Pic: https://quotefancy.com/quote/1956575/Farshad-Asl-Sharing-a-clear-and-concise-vision-spawns-a-sense-of-purpose-and-direction-It

Hands-on, exercises

- Competencies can be achieved by performing hands-on exercises
- Exercises e.g. on
  - Setting up and using test environments.
  - Testing applications on virtual and physical devices.
  - Using tools on desktops and/or mobile devices to test or assist in testing related tasks such as installation, querying, logging, monitoring, taking screenshots etc.
- Basically, hands-on learning is learning by doing

Source: https://www.istqb.org/downloads/send/61-mobile-application-testing/251-mobile-application-testing-specialist-syllabus.html
Pic and source: http://parklandplayers.com/hands-on-learning-what-does-it-mean-and-why-is-it-important/

Use all the senses

- Listen to the teacher
  - With focus!
- See the slides
  - Before, during, after session
- Talk and reflect
  - What is in it for me?
  - What is in it for us?
- Use your hands
  - Keep notes
  - Draw pics
  - Draw mindmaps

Pic and source: https://www.mindmup.com/

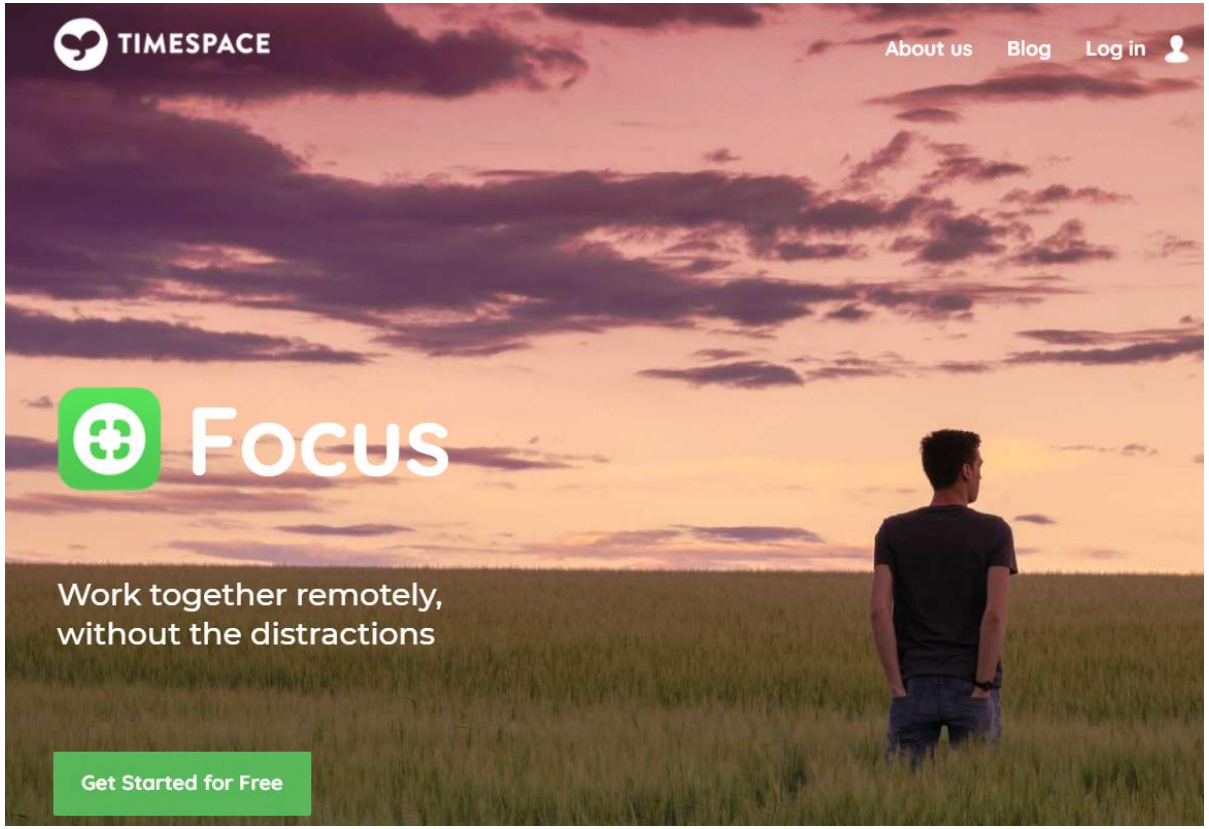
Real-time status of testing

- Use Group Memory
  - Show what is discussed in the class
  - Record working group tasks
  - Keep test plans visible
  - Track test progress in testing exercise
- Any sharing tool works
  - Mural, Miro etc.
  - Mindmaps
  - Whiteboard tools

Pic: https://www.mural.ly/

Start testing immediately

- Pick a testing approach or technique
- Apply immediately
- Use software from
  - The students
  - Your own company
  - Startup companies
- Record test progress
- Record defects
- Discuss how you think when you test



Pic: https://get.timespace.co/focus/
Source: https://we.knowit.fi/knowit-suomi/win-win-scenario-with-startups

Design your own knight (i.e. tester)

Exercise:

Design your own tester – draw your own knight

// What you need

Paper and pencil

// Task

1 **Think of a great tester you have encountered**
- Write down the characteristics of this tester to describe her/him.

2 **Think of an equivalent knight**
- Write down characteristics of the knight. If the tester is effective, maybe draw heavy armour for the knight?

3 **Draw the knight**
- Main thing is to carry through your idea of how the tester can be represented by the knight
- No need to aim for perfect picture.

4 **Add the photo to Padlet**
- Take a photo of the drawing add it to Padlet, type
- https://padlet.com/karikakkonen/qetc9f61o9vypo5h
- Or scan the QR code

Thank you!

Order the book:

https://www.austinmacauley.com/book/dragons-out

Follow and share the book project:

- https://www.dragonsout.com
- https://www.facebook.com/DragonsOutOy
- https://www.instagram.com/dragonsoutbook/
- https://twitter.com/DragonsOutOy
- https://www.linkedin.com/company/dragons-out/

Ask questions:

kari.kakkonen@dragonsout.com