

# Test Automation Pitfalls In Huge Organization

**Tuula Jokiharju &  
Tuula Pääkkönen.**

**Software Testing Days at TUT August 14-15, 2006**

# Introduction

- **Tuula Jokiharju**
  - **Specialist, SW Testing**
  - **Multimedia Computers**
  - **Needs for test automation solutions**
  
- **Tuula Pääkkönen**
  - **Global Concept Owner, Test Automation**
  - **TP/PPS/PPMS/TM Concepts**
  - **Target to offer and organize test automation solutions & services to mainly M, ES, MP and TP**

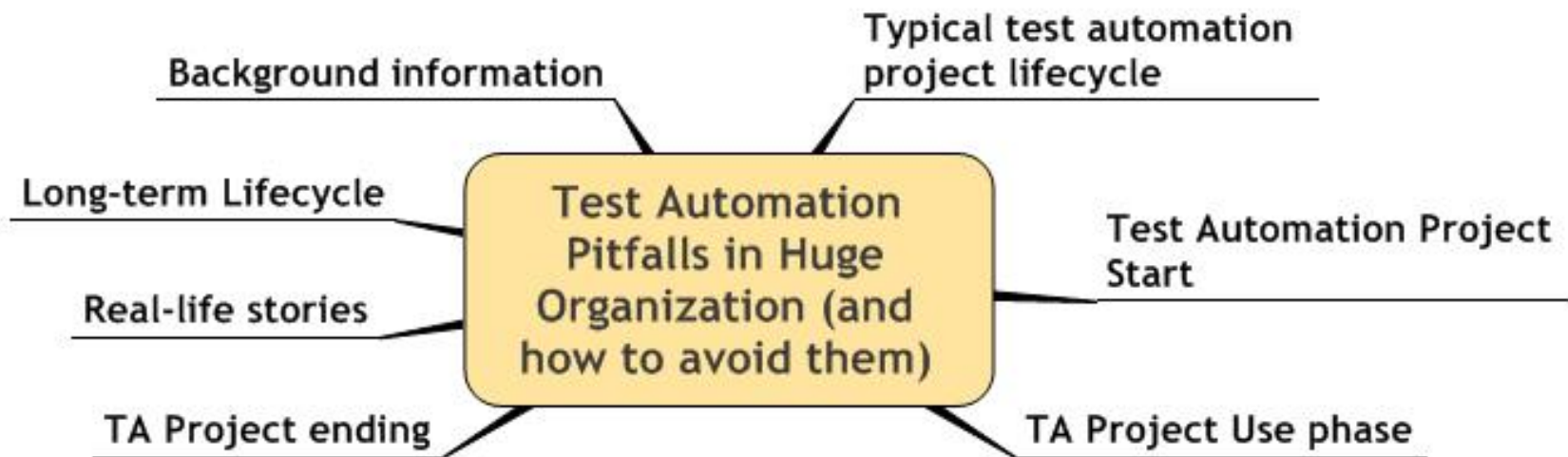


**End-user side**



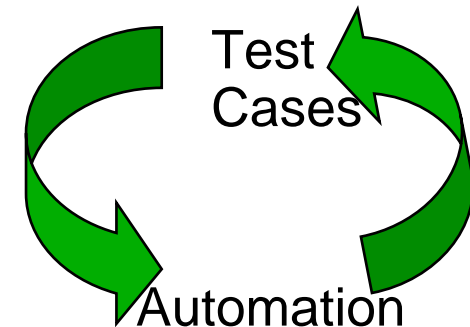
**Provider**

# Introduction of the Show

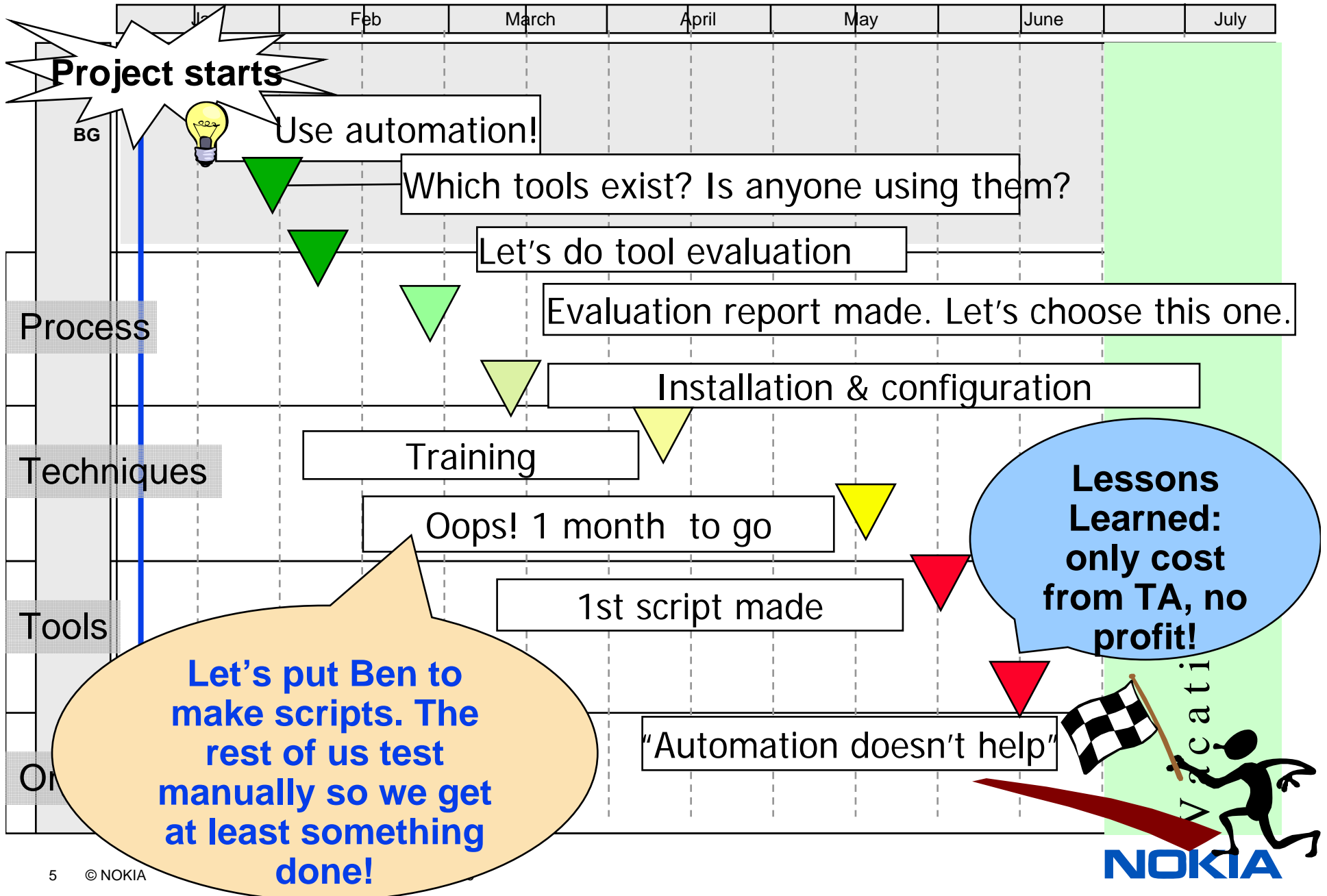


# Test Automation Briefly

- **Test automation (TA) means that you have a tool to perform or assist in certain tasks**
  - **Test execution**
  - **Comparisons**
  - **Generating test data**
  - **Record & Playback**
  - **Test management**
  - **Etc**
- **Test automation involves more tasks than just running of tool**
  - **Test Strategy**
    - **Planning what to automate**
  - **Which tool to use**
    - **GUI tool, Capture-replay, scripting languages, static/dynamic analysis tools...**
  - **How to use tools properly (scripting technologies)**
  - **Training**
  - **Support**

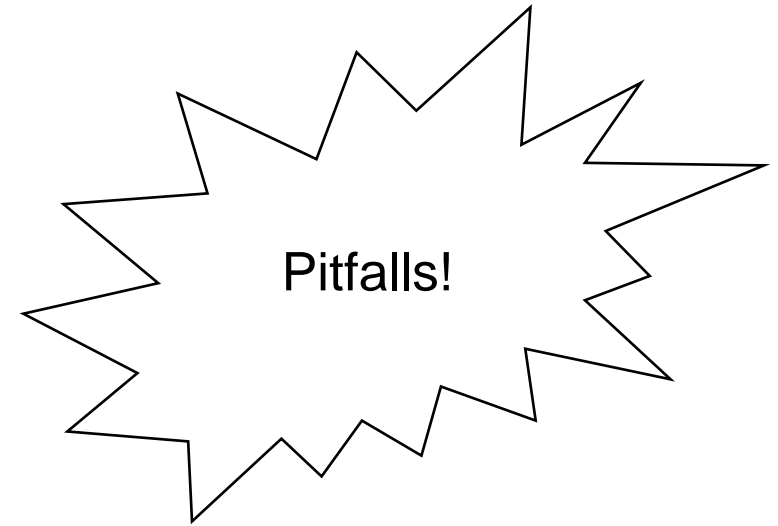


# Project



# Analysis of previous example

- **What went wrong?**
  - **TA not planned properly**
  - **Too little time**
  - **Unrealistic expectations**
  - **Not enough resources**
  - **No previous experience**
  - **Tool not suitable**
  - **Immature tool**
  - **Tool used in a wrong way**
  - **Difficult scripting language**
  - **SUT constant changes cause constant changes to scripts**
  - **Technical problems**
  - **Management support ?**
  - **...**

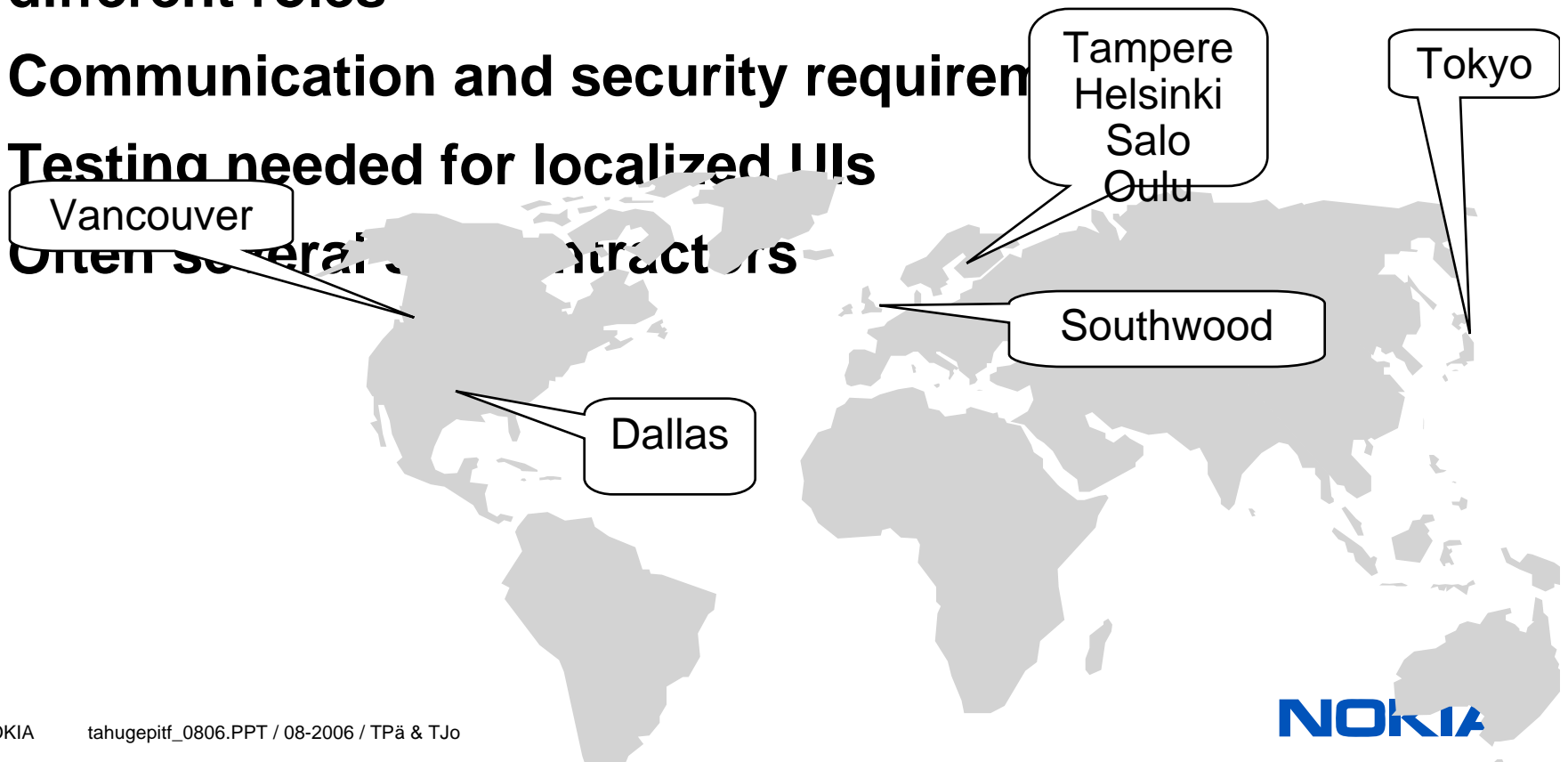


# Other pitfall origins

- Size of the organization
  - huge organization
    - sites, number of stakeholders
    - more possibilities to try-out in more versatile situation
  - tiny organization
    - limited resources (people, hw)
    - collecting competence (20% of work time of one person)
    - less possibilities in initial studies (cannot test all tools)
- Indirect pitfalls
  - general availability of existing tools
    - what exists, what exists in realistic price range, realistic expectations after demos
- Unforeseen pitfalls
  - **configuration management (CM) for scripts missing or complicated**
  - **Tool needs too much memory or storage space**

# What Is A Huge Organization?

- **Several thousands or tens of thousands simultaneous users**
- **Different time-zones, countries, languages**
- **Possibly persons from different companies in different roles**
- **Communication and security requirements**
- **Testing needed for localized UIs**
- **Often several contractors**



# Think Big!



When dealing huge organizations it has impacts...

- How global, close by is the familiar vendor? (next street/continent?)

Oulu, Southwood, Tokyo,  
Peking, Dallas,  
Bangalore, ....

- Huge corporation increases
  - Number of sites

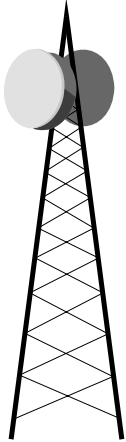
- General practices differ
- Legal and contractual issues are different
- Legacy tools in desks
- Decision makers
- Versatile needs (platforms vs. programs and different testing phases.)

- You may start to lack in
  - Awareness of global or others needs
  - Less consensus
  - What is happening elsewhere

My program rules!  
What it has in it for me?

**NOKIA**

# Impacts To TA Tools



- **Same tool suitable everywhere?**
  - **Different needs**
- **How many simultaneous connections/users possible?**
- **Retaining good service level**
  - **Tool and support should be available 24h global**
  - **Database location, backup time, recovery time and connections**
- **Training – it is quite a different thing to train 1-10 users in one site than tens or hundreds of users in various sites**
- **Planning of deployment is essential (big bang , step-wise)**
- **Computers need gigantic memory and mega speed due to amount of data**



# Avoiding pitfalls, 1st part

- **Are there previous/existing vendor contacts?**
  - **If not, how to find vendors?**
- **Do you have management support?**
  - **Money & resources reserved for the whole?**
- **Has anyone in the team any previous TA experience?**
  - **YES**
    - **Is the experience wide enough?**
    - **Is it from this tool? Do they need to unlearn?**
    - **Do they have programming/testing knowledge?**
  - **NO**
    - **A) Train yourself to understand TA and tools**
      - **Training courses, literature, e-learning**
      - **May be difficult if time is very limited**
      - **Trainees background also effects (programming, testing)**
    - **B) Use an external consultant**
      - **Are they honest and sincere, brave enough to say if you are trying to use automation in a wrong place?**
      - **Do they really understand your environment and needs?**
    - **C) Use persons from other department as consultant**
      - **Like B)**
      - **Do they really have time to help you?**

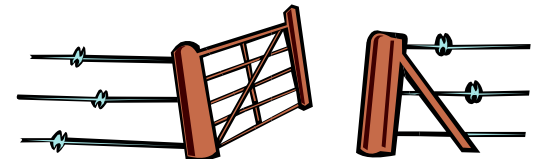
# True Story 2



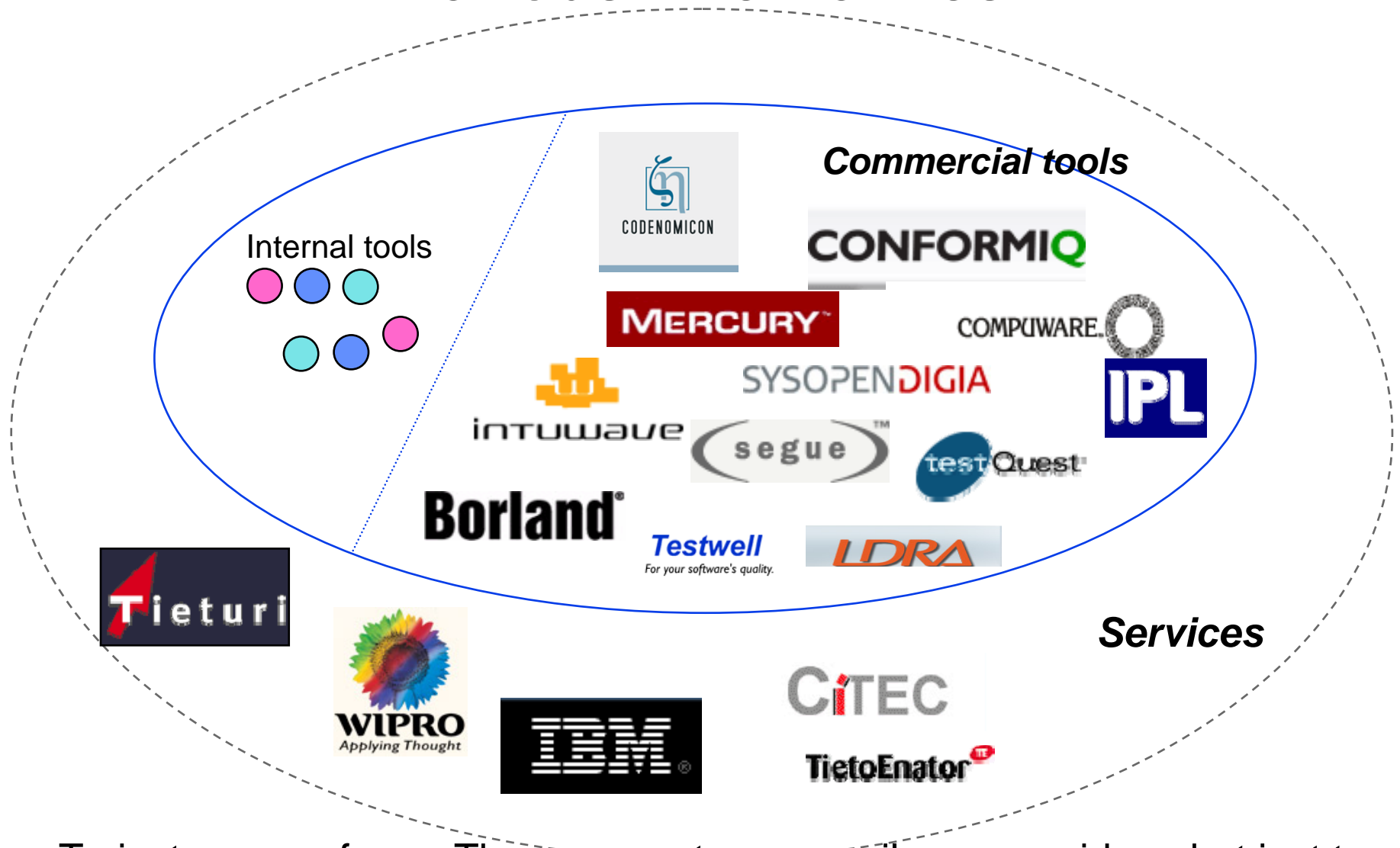
Microsoft  
PowerPoint Presentati

# Avoiding Pitfalls via “Provider” Evaluation

- There are risks with internal tools, commercial tools and open source tools
  - Be aware & proactive , plan ahead and ask questions
  - Think long term – what is needed to be able to construct TA for several years
  - Hidden costs
  - Proprietary languages within tools
  - Security assessment (what’s suitable in current situation)
  - Licensing issues
  - Test tool fence – tool learn ability and ease-of-use, [4] have different heights [1,2] in tools from different sources.
- Are there previous/existing provider & vendor contacts?
  - Utilize internal knowledge and experiences
- If not, how to find them?
  - **OVUM** vendor and tool comparisons
  - **Gartner** market analysis
  - Conferences, newspapers, people, web
  - Internal subcontracting management
- With external companies beware of
  - Red tape : various paper, legal, etc issues will take time
  - Consider both technical vs. financial aspects of vendor
  -



# Various Alternatives

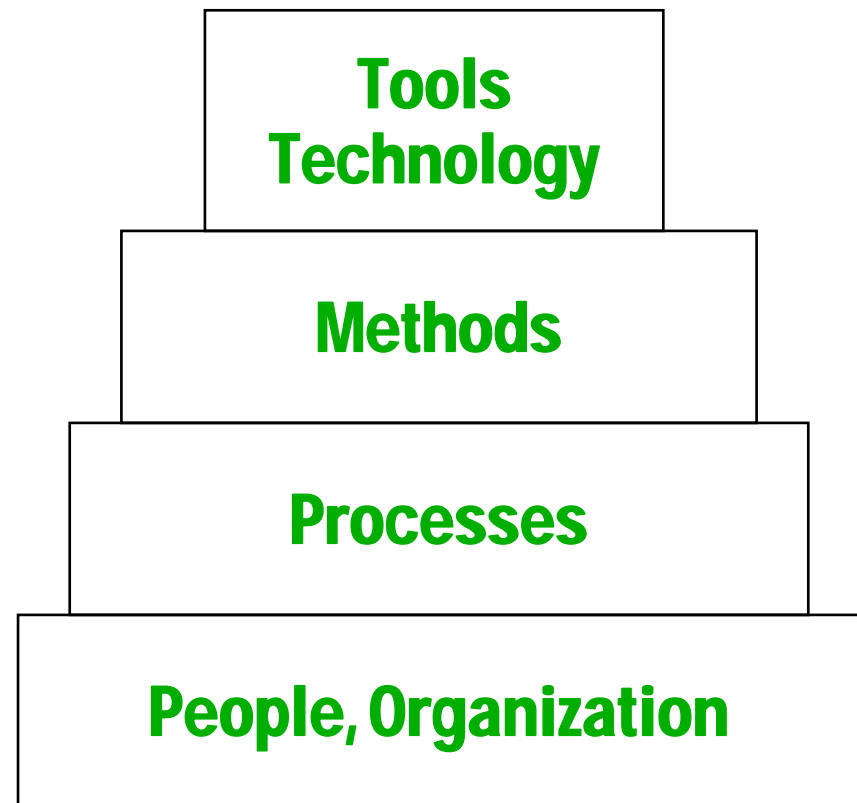
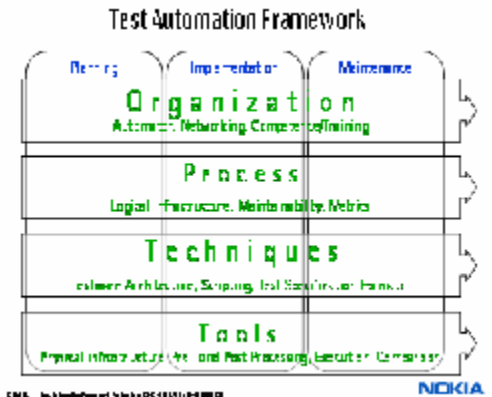
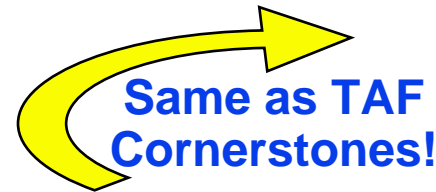


To just name a few... These are not necessarily our providers, but just to illustrate that the test tool field is very versatile.

# Tools And SW Process Aspects

original slide by Erkki Pöyhönen / TP

- Any SW process can be examined at four levels, or from four aspects:
  - People, Organization
  - Processes
  - Methods
  - Tools (technologies)
- Each change in testing practices can potentially lead a change on any of these levels
- No change on an upper level is possible without a support from below
- A visible problem on a level usually indicates a missing support from below
- A Tool is not an answer in itself



# Avoiding pitfalls #3: Tool Evaluation

- **Analyze your needs and requirements for the tool**
  - **What would be your weightings in specific program / testing phase?**
- **Don't set too high automation targets (30-40% is a GOOD level)**
  - **Be realistic!**
- **Has somebody already done this for the tools you are interested in?**
- **Demos are DEMOS: simple and chosen in a way they look fine**
- **Try with something complicated**
  - **Spend some time on evaluation**
- **Tools usually adapt to standards:**
  - **Does your application use anything SPECIAL? – They may cause a risk**
  - **Text recognition, unicode**
  - **Voice recognition**
  - **Data transfer**
  - **Non-functional requirements (reliability, stability)**

# Do Vs. Buy Decision

- In-house tools are very tempting

- Practically free
- Our own guys
- Just tuned for
- All aspects recognized
- Costs hidden



Really? (Person costs, time, hw)

**OH NO-O! Bob won 10 million in a lottery. I got an SMS saying that he bought a big yacht and is now on his way to Samoa – does anyone else know anything about the tool he's been coding???**

What about other projects?  
zed or

- Commercial tools

- Evaluation
- Costs: Buy
  - Do we
- Tool is ready for requirements
- Costs visible

**Probably not – you know he liked to work alone and he usually does not even write much comments into his code. I only know he did not get it ready yet...**

ancy, maintenance  
ning like?  
aises the



**NOKIA**

# Open source decision

- Beside the earlier mentioned building and buying the tool third option is to use open source tools
- Pros of open source
  - Generally are free
  - You can get the source code for analysis, which enables making of changes
  - There can be developers world-wide ready to do updates
- Cons of open source
  - Licensing issues might require investigating
  - How the user is going to contribute to the development
    - “polite open-source usage”
  - Can the software be used as-is, or does it require some development or tailoring
  - Securing from undesired changes of the development
  - Documentation and training (for enterprise)?

# Avoiding pitfalls , part 4

- Money matters when choosing internal, commercial or open-source tool.
- Resources and existing skills
  - Are people good at evaluating, analyzing or implementing tools?
- Company strategies and focused core competences
  - Where to put most of the effort?
  - What are the basis of strategic decisions?
  - You can't do the tool if you miss tool making capabilities
- Whatever the tool, it should just fulfill the original need
- **YOU NEED MANAGEMENT SUPPORT TO MAKE GOOD DECISIONS**
  - Concrete support needed
- And when the tool is obtained, then you need to think how to take it into use.

# Deployment

- **Plan carefully**
  - “big bang” or step-wise
  - If we take this tool into use now, what will other teams do?
- **Sell the idea of using the tool: why, why, why**
  - Prepare for change resistance
- **Guarantee enough resources**
  - Nominate key users and “evangelist”
  - Train the selected key users and end-users
- **Don’t believe you can get the moon from the sky in two weeks, but set limits to when you expect the tool to be in effective use**
- **If no visible results after several months, check the root cause:**
  - Mission not clear
  - Resources used in other activities
  - Equipment and tool do not match
  - Training missing
  - Support from Vendor not working
  - Tool no good



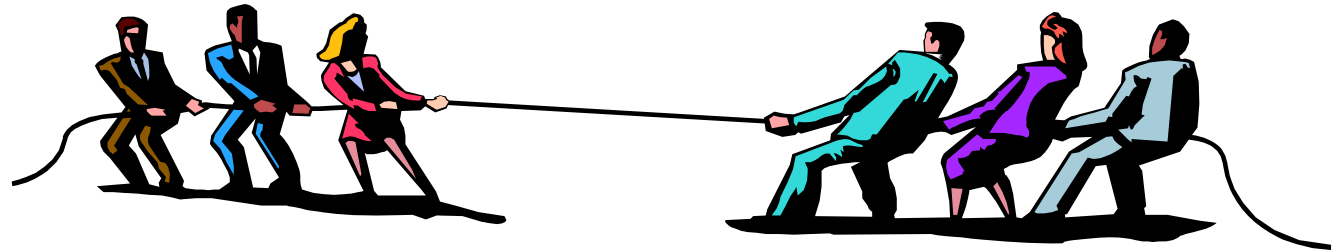
# Long-term Usage Of One Tool

- People do not want frequent changes
- Frequent changes or updates cause expenses
- How often to update versions?
  - Minor v.s. major changes
  - Backward compatibility of test scripts
- Will the characteristics of our SW remain same?
  - Will scripts be outdated because of changes?
- User amounts may increase rapidly
  - Can the used tool fulfill this demand?
  - Are there limits to simultaneous users / database size?
  - What about test case reuse and portability?
- In a huge organization database size grows rapidly
  - Do respond times remain on satisfactory level?
  - Any function to archive old data?
- Huge organization: support, process, instructions, training must be available
- Measure, does the tool really bring benefits?



# Needs Of Tool Users & Support Considerations

- **User needs**
  - Users need certain things – vendors offer certain things but the things are not necessarily overlapping
  - Even if some tool does not fulfill every tiny detail of needs, it may cover 80% of needs and **be good enough**
  - Creative users can often find several ways of using tools



- **Tool support**
  - Difficult to know beforehand what the situation will be:
  - Some vendors give good service & support
  - Some vendors are not cooperative with support
  - Some vendors have high maintenance fees
- **A huge (successful) organization has a good position to demand service**

**Exchanging Information and coordination within a huge organization is essential!**

# Long-term Maintenance & Support

<b>Changes do happen</b>	<b>How to control the change?</b>
<b>Our products change</b>	<b>Tool adaptation to new features, platforms and versions proactively</b>
<b>Infrastructure changes (operating systems....)</b>	<b>Roadmap changes with vendor, communicate</b>
<b>Other tools will change</b>	<b>Integrations, interfaces, conversions, wrappers</b>
<b>Different users, more users</b>	<b>Training path, delta trainings, gathering new requirements &amp; analyzing user feedback</b>
<b>Vendor status might change changes in financial situation (merges, acquisitions...),</b>	<b>Vendor relationships closer, alternative solutions</b>
<b>Tool data gets outdated, amount of data grows</b>	<b>Archiving (what to archive, when), transferring, redoing</b>

# Ramp-down

- **Tool ramp-down may be a great loss or a dream come true**
- **Reasons for ramp-down**
  - **The system SW changes and the tool is no longer integrable**
  - **The nature of our SW or testing purposes change and the tool cannot be used anymore**
  - **The amount of users increases and we reach the limits – the tool causes much annoyance**
- **Postponing ramp-down**
  - **You can make rules to limit simultaneous use and clean databases**



# Possible Difficulties in Ramp-down

- In a huge organization changing any tool may need a massive amount of careful planning and extra work
- Is there any chance in the future to use the huge amount of information possibly stored in the tool's database
  - Can the information be converted to the new tool?
  - Which information to convert?
- Change resistance
  - Why change working tool?
  - As in deployment, answering to “Why?” helps
- Hidden integrations and interfaces
- Process changes are possibly needed
- Unlearning old practices



# True Story 3



Microsoft  
PowerPoint Presentati

# What's so different between small and big organization?

- Big is more often located in several countries
- In small organization one person has several hats.
- The number of simultaneous users effects service levels of tools and requires more complex tools.
- In big organization all services cannot be local, due to data transfer rates and simply server management.
- Minimizing downtime and maintenance breaks
- Big organization requires more formalism by defining services
- In big organization harmonization of tools, processes requires effort.

# Summary

**Decide**

**Identify**

**Plan**

- **Decide on TA strategy**
- **Identify your needs**
- **Plan, communicate and follow-up**

**Define**

**Introduce**

**Pilot**

- **Define reasonable goals**
- **Introduce resources, support and training**
- **Pilot must not be too simple or easy**

**Deploy**

**Increase**

**Prepare**

- **Deployment planning is needed**
- **Increase gradually the scope in a huge organization**
  - **Think globally, not just your own tiny project**
  - **Some compromise may be needed**
  - **A tool can be good enough even if it is not optimal**
- **Prepare to long-term tool usage, and plan ahead for ramp-down and archival phases, also**

# References

- [1] Steve Morton, Understanding Both Sides of the Test Tool Fence, [http://www.stickyminds.com/pop\\_print.asp?Object\\_id=3083&ObjectType=ART](http://www.stickyminds.com/pop_print.asp?Object_id=3083&ObjectType=ART)
- [2] Gartner, Magic Quadrant for Application Quality Ecosystem, 2005: Leaders and Challengers
- [3] Kaner et al. , Lessons Learned in Software Testing
- [4] Automated software testing, Elfriede Dustin, Jeff Rashka, John Paul. ISBN: 020143870
- [5] web articles: [www.satisfice.com](http://www.satisfice.com) [www.stickyminds.com](http://www.stickyminds.com).  
[www.testingcraft.com](http://www.testingcraft.com)

QUESTIONS ?

THANK YOU

