# Ten Tips for Home-Baked Tools

Kathy Iberle

Rose City
Software Process Improvement Network
Portland, OR
June 2007

# *Cautionary Tales from the Trenches*

Tales of mishaps and mayhem…

- theme: writing software development tools
  - configuration management, defect tracking, test case management, etc.
- characters: people who are writing a tool for their own use
- settings:  a variety of projects and companies over the past twenty years

# Once upon a time, someone said….

- *“Doesn’t every programmer write a defect tracking system sooner or later?”*
- *“We just need a list of requirements”*
- *“SQL is just another programming language”*
- *“It’s just a small program - we can put it on the machine under my desk”*
- “*A couple of people could do that in a month”*

***“Every programmer writes a defect tracking system sooner or later”***

## *“Every programmer writes a defect tracking system sooner or later”*

- The tale:
  - The developers work for a small start-up, and are just out of college
  - Need a defect tracking system which can be accessed by multiple companies across their respective firewalls
  - Decide to adapt BugZilla
  - All goes well for awhile….

## *“Every programmer writes a defect tracking system sooner or later”*

- The tale:
  - The developers work for a small start-up, and are just out of college
  - Need a defect tracking system which can be accessed by multiple companies across their respective firewalls
  - Decide to adapt BugZilla
  - All goes well for awhile….
  - Until we demand trend metrics
  - Which have to be created by hand…

## *"Every programmer writes a defect tracking system sooner or later"*

- Tip #1: Count *all* your costs
  - What will it cost to build ***and maintain*** your home-baked tool?
  - What features will be needed that it hasn't currently got?
- Tip #2: Understand your context
  - Cheapest may not have enough power
  - Biggest is not always best

## *“We just need a list of requirements”*

- Go ye forth and obtain a bulleted list of features...
  - “Everybody knows this is the way to start a project.”

## *“We just need a list of requirements”*

- The tale (or, rather, several tales)
  - The administrator of a related existing tool is charged with collecting requirements.
  - Users are interviewed and a list of features is collected.
  - The team implements every item on this list.
  - The tool is released to its users….

## *“We just need a list of requirements”*

- Surprise!
- The tool doesn’t do the job adequately!
  - Common use cases can’t be performed.
  - OR
  - Some users are happy and others unhappy.

## *“We just need a list of requirements”*

- Choose your own ending…
  - More features are added until all the users can use it, but now the tool is confusing and costs a lot to maintain.
  - The different user classes fight with each other over whether the tool is “acceptable” until the project dies.
  - The sponsors give up and go shopping.

# *"We just need a list of requirements"*

- What is going on
  - Use cases are missing.
  - Some user classes are missing more use cases than others.
- Why does this happen?
  - perception that software development starts with a list of requirements
  - don't realize there is a prior business analysis step
  - In the last 10 years, with the popularity of agile methods, I see a lot more developers aware of the business analysis step.

## *“We just need a list of requirements”*

- Tip #3: Start with use cases
  - Use cases will reveal the too-obvious-to-mention tasks
  - It’ll be more obvious when different groups of users have different use cases.
    - Different data
    - Different set of actions
- Lists of requirements can be generated from the use cases.

## *“We just need a list of requirements”*

- Tip #4: Use models in addition to lists and use cases
  - For complex processes, try a swim-lane diagram or state charts
  - For complex data, use data models
  - Models help ascertain whether variant use cases must be variant

## *"We just need a list of requirements"*

- The moral of the story:
  - Designing a major information management tool requires systems analysis skills
    - System administrator is not guaranteed to have them
    - Users are not guaranteed to have them either
  - Get at least one person on the team who does have these skills
    - or is willing to learn them

# *"SQL is just another programming language"*

- The tale:
  - Talented self-taught programmer creates a great tool
    - No off-the-shelf tool comes close to meeting the business need

# *"SQL is just another programming language"*

- The tale:
  - Talented self-taught programmer creates a great tool
    - No off-the-shelf tool comes close to meeting the business need
  - But, eventually one of these things starts happening:
    - Performance goes downhill rapidly as usage grows
    - Upgrade to a later version of the dbms crashes
    - Functionality cannot be extended for a reasonable cost

# *“SQL is just another programming language”*

- Why?
  - Fundamental design problems
  - Developer(s) did not know some of the basics of database-backed system design
  - Developer(s) did not know how to use a relational dbms

# *"SQL is just another programming language"*

- Tip #5: If you need a relational database in your tool, you need someone who knows databases
  - Either get a developer trained in these specific skills
  - Or send one of yours to class
  - It will pay off.

# *"SQL is just another programming language"*

- Another common chapter:
  - Difficult for anyone other than the original author to work on the tool
    - Business logic is mixed into the UI code
    - Didn't separate the logical layers:  data, business logic, user interface

# *“SQL is just another programming language”*

- Tip #5: If you need a relational database in your tool, you need someone who knows system design

# *"SQL is just another programming language"*

- The moral of the story:
  - Brilliance in one field of software development doesn't guarantee automatic brilliance in another

# *"We can put it on the machine under my desk"*

- The tale:
  - We need a tool and we need it now
  - I have this extra machine
  - We'll load the tool on this machine
    - Might not even be a home-baked tool, but it's home-baked IT support
  - Everything goes all right for awhile

# *"We can put it on the machine under my desk"*

- But then…
  - The machine goes on the fritz
  - Start looking for a backup machine…. a backup of the data ….

## *“We can put it on the machine under my desk”*

- No machine is invulnerable
- You’ll need a regular backup
- You’ll need some sort of security
- The network might change

# *"We can put it on the machine under my desk"*

- Tip #7: Plan support for the back-end machine
  - Backups and disaster recovery
  - Security
  - Performance under load
- At the very least, get advice from your organization's IT department
  - (not the help desk, the people behind them…)

# *"We can put it on the machine under my desk"*

- The sequel to the tale…
  - The tool becomes ever more popular as the project grows
  - Now there's 20 users, not 5 users
  - And 8 of them are in India
  - The author is answering phone calls about the tool during the day
    and at night…

# *"We can put it on the machine under my desk"*

- Tip #8: Plan for user support
  - What support will you need?
    - Help for users?
    - Someone to perform administrative actions?
    - Does the support need to be 24x7?
  - Set up user support **before** the users are driving you crazy

# *"A couple of people could do that in a month"*

- The oft-repeated tale:
  - The tool looks easy enough
  - Team starts to collect requirements
  - Discovers that there are a lot more use cases than they thought

# *"A couple of people could do that in a month"*

- Multiple unhappy endings:
  - Project runs way over budget,
    and there is much recrimination
  - Project delivers on original schedule but half the users can't use the tool,
    and there is much recrimination
  - Project is cancelled,
    and there is much recrimination

# *“A couple of people could do that in a month”*

- Tip #9: Estimate, estimate, estimate
  - Estimate your internal projects the same way you estimate your “real” projects
  - If you don’t do software development projects, borrow estimation processes from a group that does

# *“A couple of people could do that in a month”*

- Another story:
- The configuration management project
  - had to handle 600MB files
  - no off-the-shelf tool within budget
  - decided to use a variation of eXtreme Programming

# *“A couple of people could do that in a month”*

- 1st iteration:
  - no coding at all
  - simply a written set of procedures for using existing shared disk space

# *"A couple of people could do that in a month"*

- 1st iteration:
  - no coding at all
  - simply a written set of procedures for using existing shared disk space
- 2nd iteration:
  - implemented top 8 user stories
  - did not implement "tagging" or "labeling" (not in the top 8)
  - 300% over resources budget, but…

# *"A couple of people could do that in a month"*

- 1st iteration:
  - no coding at all
  - simply a written set of procedures for using existing shared disk space
- 2nd iteration:
  - implemented top 8 user stories
  - did not implement "tagging" or "labeling" (not in the top 8)
  - 300% over resources budget, but…
  - Happiest users I've ever had

# *"A couple of people could do that in a month"*

- Tip #10: Try an agile development method
  - Basic use cases now is better than a full set of use cases much later
  - Project can be stopped at many points without losing the value of what has already been done

# Conclusion

- Tip #1: Count your costs
- Tip #2: Know your context
- Tip #3: Start with use cases rather than bulleted lists of requirements
- Tip #4: Use models in addition to lists and use cases
- Tip #5: If you need a relational database in your tool, you need someone who knows databases
- Tip #6: If you need a relational database in your tool, you need someone who knows system design

## Conclusion

- Tip #7: Plan support for the back-end
- Tip #8: Plan for user support
- Tip #9: Estimate, estimate, estimate
- Tip #10: Try an agile development method

# *Questions?*

- Slideset will be available at www.kiberle.com after the SPIN presentation on Jun 14.
- Email me at kiberle@kiberle.com