Outcomes

- Looks really cool
- Fits like trousers
- Floats
- Rolls like a whirling dervish
- Acquired skills
- Appreciated the role of the expert & apprenticing
- Experience shared & magnified as a group
- I love it!
My Critical Insight

- Requirements for a personal kayak are easy to determine once you get to the heart of what you want & why
- Paring down to those essentials however, in a world of endless possibilities & dreams, is the really hard part
- I got what I wanted only by letting go of what I did not really need
- The requirements evolved… quite naturally… as part of a dialogue … thanks to a good process
But…

I had **domain** knowledge & expertise

Trade-offs & decisions were felt **personally**
RE’s BIGGEST Challenge

- The problem of achieving an agreed set of requirements amongst a set of stakeholders was clear in this context -- we could NOT have built a kayak to suit all of us.

- This issue remains one of the largest & underappreciated challenges of software requirements engineering.

- The only way in which this is sometimes possible is to build a software system that has compromised the requirements away.
Checklist

1. Rip a story stick to serve as a baseline -- Know your top level goal & key parameters to always work within

2. Agree on terms -- Agree on terms (domain & engineering) & provide a project glossary

3. Make marks consistently -- Use standardized conventions to communicate & record necessary work

4. Proceed from an architectural backbone -- Be clear about those requirements that are stable and shape the architecture; care for them

5. Envision prior to any commitment -- Take time to understand the global impact of local changes & decisions
Checklist

6. Measure twice, cut once ... revisited -- If you are truly engineering, take time to measure?

7. Stabilize to localize change & care for what remains -- Structure your requirements, give them identifiers & attend to flow-down for traceability

8. Work within acceptable tolerances -- Know your degrees of freedom at each step of the RE process

9. Avoid single points of failure & unnecessary rework -- Start to learn through accumulating lessons & best practice

10. Employ models & create informal sketches where useful -- Don't communicate in an alien tongue & use props where possible
11. Understand the role and value of the expert -- Ensure people learn on the job from inspiring masters

12. Know the most precious resource & the most ubiquitous -- You DO need more than people & money, so think about total resourcing

13. Use the right tool for the job -- Buy tools with care, they can be your worst investment

14. Master the underlying techniques to capitalize upon use of the tool -- Teach fundamental skills & always put process first

15. Embrace diversity & pull together -- Remember that the team works better if each member can champion & be proud of something
Checklist

16. Provide for a healthy & safe working environment -- If you don't care for those on your project, why should they care about the project?

17. Recognize that the best is the enemy of the good -- Know what is 'good enough' & know when you are done

18. Watch that urge for closure -- Plan to keep the requirements in mind right through the final hours

19. Conduct early testing & a final fairing -- Use the requirements to test every step of the way

20. Determine the risks & have a back-up plan -- Do this as best you can, but sometimes you just can't control the weather... Us Brits have been using that excuse for years!!!
For More…

- Read my 2-part article
  Building Myself a Kayak: Some Lessons for Requirements and Software Engineering

Requirements Quarterly
Issues 53 & 54
January 2010 & April 2010

The Requirements Engineering Specialist Group of the British Computer Society
(http://www.resg.org.uk/)

- Contact: ollly@gotel.net