(Not) Playing with Trains: a real-time safety-critical development with USDP

zühlke DIE DENKFABRIK.

Rational the software development company

Jake Stewart

Principal Technical Architect
Zuhlke Engineering Ltd
jcs@zuhlke.com

The System Engineers: Zuhlke

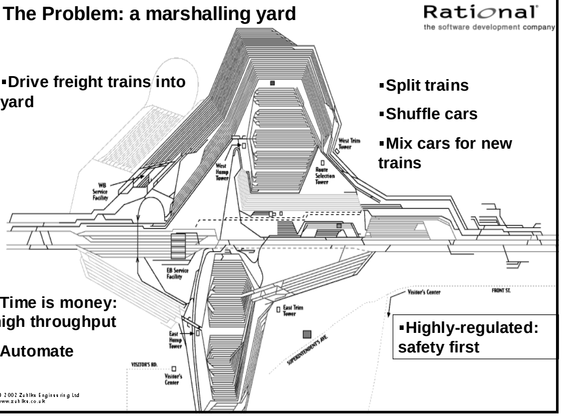
- Zuhlke is a leading Swiss technology consultancy. We specialise in the leading edge technologies required in today's enterprise systems, and in the processes and methodologies required to engineer them.
- Our core technology competencies are in component and object based systems, security, distributed architecture; our process competencies are in iterative development and project management.
- We have clients ranging from Alstom and British Airways to Xerox and Zurich Financial Services, and operations across Switzerland, Germany and the UK.
- In our 34 year history, we have completed over 3000 projects for over 500 clients. 85% of our business is from existing clients. Our first customer – Gretag, in 1968- is still a customer.

zühlke

The Problem: a marshalling yard

Rational the software development company

- **Drive freight trains into yard**
- **Split trains**
- **Shuffle cars**
- **Mix cars for new trains**



- **Time is money: high throughput**
- **Automate**
- **Highly-regulated: safety first**

The Project: software based control

Rational the software development company

Software Needs

- Extensive domain functionality
- Real-time control of train movement
- Distribute logic to smart railyard components
- Features and process must fulfill safety-critical classification

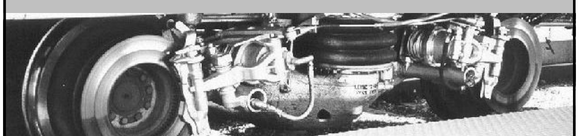
zühlke

The Project: a drilldown example

Putting on the brakes

Rational the software development company

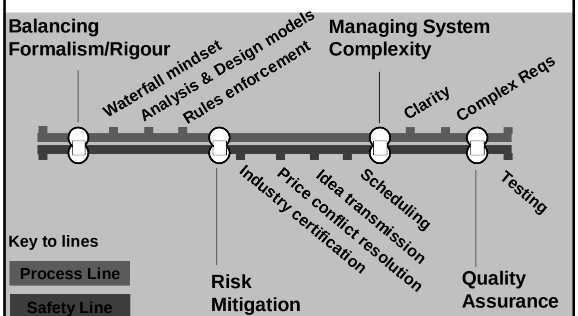
- Moving train cars must be linked up
- Must control speed without their brakes
- Stationary brakes along track
- Special Risks
  - Collision (under- or over-braking)
  - Derailment (mis-braking so wheels jump out of brake shoe and off track)
- Ensuring safety with process



zühlke

Our Journey Planner

Rational the software development company

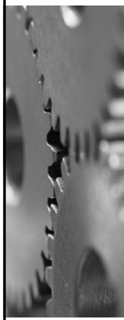


zühlke

Our Journey: Process and Safety
What is special about safety-critical systems?

- **Safety-critical systems may be defined as systems whose incorrect operation can result in:**
  - Human injury
  - Loss of human life
  - Grievous injury to the corporate bottom line
    - (depending on who's doing the defining)
- **Emphasis on:**
  - Hazard analysis
  - Correctness
  - Formal process
  - Risk mitigation

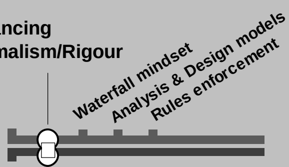
Our Journey: Process and Safety
Which process to use?



- **The debate (1): Iterative or Not?**
  - "You need a serious methodology for safety critical"
- **The debate (2): RUP**
  - "RUP is too lightweight for 'serious' projects"
  - "RUP is too heavyweight and restrictive"
- **Our observation**
  - RUP has significant natural alignment with safety-critical development
  - But there are aspects of safety-critical development that are not covered

Next Station: Balancing Formalism/ Rigour

Balancing Formalism/Rigour



Finding a Balance of Formalism / Rigour

Higher formalism is appropriate on a safety-critical development

- **Upsides**
  - Thoroughness
  - Traceability (aided by good process tool)
  - Clear audit trail (mandatory in safety critical)
- **Downside**
  - More bulk, more admin, longer timeframes
- **Select and adapt your process aids carefully:**
  - RUP is *intended* to be adapted

Finding a Balance of Formalism / Rigour
Avoiding a waterfall mindset

- **High-formalism beneficial practice:**
  - Separating teams along phases builds design sensibility and documentation sufficiency
- **Pitfall:**
  - "Use case assembly line" is lossy and encourages design-by-analyst
  - Waterfall mindset
- **Remedies**
  - Internal iteration with arch/design reps before analysis handoff
  - Handoff focus should be model clarity, not size

Finding a Balance of Formalism / Rigour
Analysis and design models

Maintain separate analysis model or not?
- RUP leaves it open

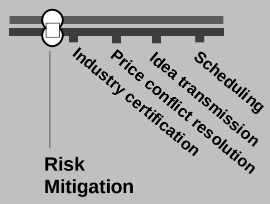
- **Some prefer them**
  - Bridge between business domain and implementation
  - Clear, pure view of concepts
- **Some don't**
  - Nightmare keeping consistent
  - Or worse
- **I don't**

Finding a Balance of Formalism / Rigour
Rules enforcement

- **Rules are good...**
  - No slipping through the cracks
  - Ensure quality if (when) "the crunch" comes
- **...but you must always rule the rules**
  - Experienced leader in control of the rules
    - They can adapt the rules if necessary
  - Keep brains turned on behind the process

MIND THE GAP

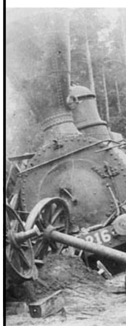
Next Station: Risk Mitigation



Risk Mitigation

- **For safety-critical system, risk mitigation is king**
- **Ditto for RUP practices**
  - Risk mitigation drives development process
  - Encouragement to unearth, & publish risks

Risk Mitigation
Industry safety certification



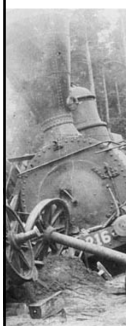
- **Industry certification may be mandatory**
- **Methodical risk-elimination is cornerstone of safety certification**
  - European and local-country oversight
  - Certification prescribes practices:
    - Risk & Hazard Analyses
    - Requirements specification, quality assurance
      - Graphical description
      - 6-eyes principle
    - Design and Programming
      - Defensive programming
      - Error recognition and diagnosis
  - Many standard RUP processes are certification-friendly

Risk Mitigation
Price conflict resolution

- **Systems that put human life and/or substantial commercial assets at risk (safety-critical) pose tough development challenges**
- Pushing costs up
  - Verifying correctness and building highly-robust systems
- Wanting costs down
  - Can force a conflict with maintaining correctness
- **This can lead to spectacular failures**

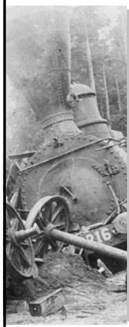
  - RUP offers some edges on this problem.

Risk Mitigation
UML and idea transmission



- **UML as common written language**
  - Becoming a widely trained skill
  - Helps toward error-free transmission of ideas
  - Minimises opportunities for misunderstanding
- **But watch for "Pidgin-UML"**
  - Counter-productive at best
  - Dangerous at worst
  - Enforce a minimum working standard

Risk Mitigation
Scheduling



▪Ensuring higher level of quality needs time

- **Plan a longer timeframe**
  - Allow time for chosen process activities
  - Allow project plan to evolve
  - Schedule more iterations
- **Allow results of iterations to shape project plan**
  - Especially rather than vice versa!
- **Take advantage of intra-iteration cycles**

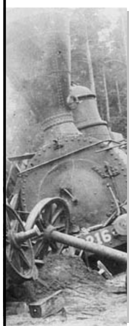
Risk Mitigation
Outside of RUP

▪RUP helps with 'how' to build

▪Not 'what' to build for safety-critical

▪Appropriate supporting architecture
- Selection comes from experience and domain practices
- Standard safety design patterns
  - Keyed watchdog
  - Monitor actuator
- Fail safety
- Physical and algorithmic redundancy

Risk Mitigation
Outside of RUP



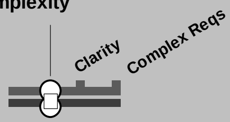
▪RUP guides a team, but can't solve issues or give experience to the team

▪RUP provides...
- Benefit of packaged collective experience of others
- A map to tie together certification requirements

▪But the team are still...
- Responsible for domain experience, best practices, design decisions, etc
- There is not yet found a process that gets the same results from a mediocre team as from a fantastic team

Next Station: Managing System Complexity

Managing System Complexity



"But this system *is* complex!": Managing Industrial-Strength Complexity

▪Rule of thumb for maintaining quality when designing system is keep unnecessary complexity out of the models

- *"That's all well and good for some example system. I have real-world requirements to capture, though!"*
- *"But this system is complex! How can I just 'reduce complexity?'"*

▪On non-trivial systems, this is easier said than done

"But this system *is* complex!"
Clarity

Models

▪Unclear
- Too-detailed or too many hinders understanding
- Must never obscure the concepts being illustrated
- Unusable models negate good individual elements

▪Clear
- Filter, to be accurate without being overwhelming
- Architect and PM keep model scoping on track

“But this system *is* complex!”
Clarity

Rational
the software development company

It is hard work to decide what is (not) significant

- **Often interests can seem at odds:**
  - E.g. “Airplane Rule”
    - “ Complexity increases the possibility of failure. A twin-engine airplane has twice as many engine problems as a single-engine airplane. ”
    - Hence?

zühlke

“But this system *is* complex!”
Clarity

Rational
the software development company

Subjectivity Needed

- **Arch/designer choose the “interesting” parts to illustrate**
  - No “every-detail” diagrams
  - Forced to decide what *is* important
- **A computer could easily auto-doc *everything***
  - But overwhelm the salient parts
  - Also introduce redundancy

zühlke

“But this system *is* complex!”
Clarity

Rational
the software development company

- **Clarity of any part of the design is constantly critical**
- **UML adds value again**
  - A set of standard notations
  - Increases visibility and transparency
    - Across all project team roles
    - And through all phases of the project
  - Specialization for real-time concepts

zühlke

“But this system *is* complex!”
Complex requirements

Rational
the software development company

Analysing complex business requirements

- **Benefit**
  - Deep analysis by business experts roots out idiosyncrasies
- **Pitfall**
  - Deeply-organized business requirements may be at odds with best software organization

zühlke

“But this system *is* complex!”
Complex requirements

Rational
the software development company

- **Business requirements as pseudo-implementation are too deep**
  - Further changes feel “too late”, “not allowed”
  - Too much inertia
- **Better approach:**
  - Designers identify better organization
    - Real-time confirmation from business side
  - Do encourage business experts to go deep
    - Nip unproductive tangents in the bud
- **Realization should flesh out specs, not contrast**

zühlke

“But this system *is* complex!”
Complex requirements

Rational
the software development company

- **Key observations**
  - Use appropriate strengths of appropriate artifacts
  - Insert reference/pointer from reqs docs into relevant spots in design
  - Don’t insulate designer/coder from raw requirements
  - Don’t restrict the view up- and downstream

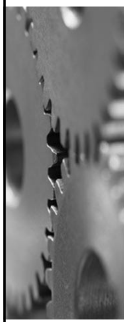
zühlke

Next Station: Quality Assurance

Testing

Quality Assurance

Quality Assurance

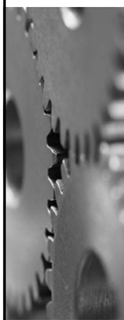


- Quality Assurance Processes are mandatory
- Standard processes, but heavier emphasis
  - A "larger than usual" allocation of resources and time
  - Flexible team structure
    - But all know their roles
      - e.g. subsystem can't make it into coding until team of project architects has reviewed architecture/design
  - Guidelines, reviews
  - Testing, testing, testing

Quality Assurance
Testing

- Actual environment testing may be expensive
  - Good simulation a necessity
  - Simulate from every relevant angle
- Incorporate parts of actual hardware early in the process
  - Adjust designs based on early-discovered hardware surprises

Quality Assurance
Testing

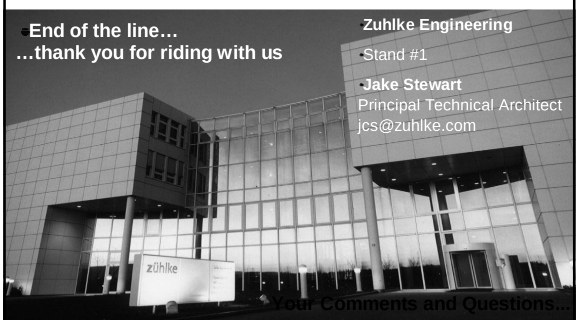


- Tools
  - Multiple commercial tools available
  - Rational testing tools
    - Environment integration may be a benefit
- Address specific risks of system
  - Best solution may be combination of commercial and customized
  - Most valuable test data came from separate test strategies

Summary

- **RUP was an asset to us with large real-time safety-critical development**
- **Easy to see why**
  - Risk-mitigation-centric
  - Overlaps with and enhances safety certification requirements
  - Shows roadmap through overload of extreme functional and non-functional reqs
- **A good process drives a project**
  - **Care must be given to driving the process, too!**

DIE DENKFABRIK.



- End of the line…
…thank you for riding with us

Zuhlke Engineering

- Stand #1

- **Jake Stewart**
Principal Technical Architect
jcs@zuhlke.com

Your Comments and Questions...