

Agile Development

Trials and Tribulations

Horia Slusanschi, PhD
Monday 26 July, 2004

- > **Systems Integration.**
- > **Outsourcing.**
- > **Infrastructure.**
- > **Server Technology.**
- > **Consulting.**

UNISYS

Imagine it • Done •

Topics

> **Agile & Iterative approaches vs. Status Quo**

- Brief overview of the Agile Manifesto
- The Spirit of RUP
- The contrast of the Waterfall and Iterative sweet spots

> **Business Trials**

- Budgeting strategies
- Collaboration challenges

> **Software Engineering Tribulations**

- Documentation? What documentation?
- A hitchhiker's guide to the UML Fevers

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- > Individuals and interactions** over processes and tools
- > Working software** over comprehensive documentation
- > Customer collaboration** over contract negotiation
- > Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

<http://www.agilemanifesto.org/>

UNISYS

Imagine it • Done •

RUP — Iterative Development

> RUP phases

- Initiation
- Elaboration
- Construction
- Transition

> RUP core practices

- Develop Iteratively
- Continuously Verify Quality
- Use a Cohesive Component Architecture
- Manage Requirements
- Manage Change
- Model Visually (UML)

The Spirit of RUP — Kruchten (I)

- **Attack major risks early and continuously, or they will attack you**
- **Ensure that you deliver value to your customer – early and often**
 - Do nothing useless (that does not deliver value to either the client or your team)
- **Stay focused on delivering executable software in early iterations, not specifications or other documents**
- **Accommodate change early in the project through provoking and managing change via early development**

The Spirit of RUP — Kruchten (II)

- > **Create an executable, cohesive architecture early on**
- > **Build your systems with components**
 - At all times seek to reuse wherever possible
- > **Work together as one team – e.g. cross-functional teams**
- > **Quality is a way of life, not an afterthought**
 - Insist on quality in each action
 - Never expect that “someone else” will eventually “tidy-up”
 - Re-factor judiciously and constantly for simplicity and quality

Waterfall

> Oil production

- Deposit layout & capacity known
- Extraction requirements well-defined

> Industrialised

- Very precise requirements & specifications
- Predictable
- Repeatable

> Work in isolation

- Stable process, with clearly defined roles and responsibilities
- Hierarchical organisation

Iterative & Agile

> Oil exploration

- Unknown deposit layout & capacity
- Unknown extraction requirements

> Custom-made

- Evolving requirements & specifications
- Adaptive
- Flexible

> Collaboration

- Roles and responsibilities are fluid, as the team adapts and learns
- Self-organising teams

Sweet Spots

> Waterfall — Process focus

- Existing technology
- Well-defined, unchanging context
- Repetitive, well known tasks
- Teams with set, immutable roles and structure

> Iterative & Agile — People focus

- New or existing technology
- Exploratory, fluid context
- Adaptive, unexpected or previously unknown tasks
- Self-organising teams, dynamic roles and structure

Business Trials

The Rock

- > Systems Integration.
- > Outsourcing.
- > Infrastructure.
- > Server Technology.
- > Consulting.

UNISYS

Imagine it • Done •

Budgeting Strategies

> Typical approach

- Cost it
- Get budget for it
- Build it (hopefully within cost)
- Test it (if time allows)
- Figure out what 'it' actually is, and whether it's any use

> Alternative

- Budget for a time-box
- List lots of useful features
- Pick the most [business] rewarding and [technically] riskiest features that will fit in the time-box
- Deliver them by the end of the time-box
- If the business objective has been reached, stop.
- If not, repeat from the start

Collaboration Challenges

> Office layout

- Desk ergonomics for collaborative work
- Caves & common
- Whiteboard surfaces on the walls for modelling

> Habits

- Can-do attitude
- Continuous learning
- Communication skills

> Tools

- Model-driven development tools
- Digital camera to take whiteboard snapshots
- Plotter for displaying large models on walls

> Personal hygiene

Software Engineering Tribulations

The Hard Place

- > Systems Integration.
- > Outsourcing.
- > Infrastructure.
- > Server Technology.
- > Consulting.

UNISYS

Imagine it • Done •

Software Development — A Game

- Software development is a series of resource-limited, goal-directed cooperative games of invention and communication.
- The **primary goal** of each game is the production and deployment of a software system; the **residue** of the game is a set of markers to assist the players of the next game. People use markers and props to remind, inspire and inform each other in getting to the next move in the game.
- The next game is an alteration of the system or the creation of a neighbouring system. Each game therefore has as a **secondary goal** to create an advantageous position for the next game.
- Since each game is resource-limited, the **primary and secondary goals compete for resources.**

Alistair Cockburn, Cockburn Cooperative Game Manifesto

Documentation? What documentation?

> Resource competition

- Primary goal: Production and deployment of a software system
 - The only mandatory RUP artefact is code
- Residue: Set of markers to assist the players of the next game
 - Automated testing frameworks, sketches
- Secondary goal: Create an advantageous position for the next game
 - Capture tacit knowledge and the oral team culture as documentation

> Documentation is a Business Investment Choice

- Code structure and documentation as technology “loans”
- If careless, “interest payments” may become overwhelming

UML Fevers

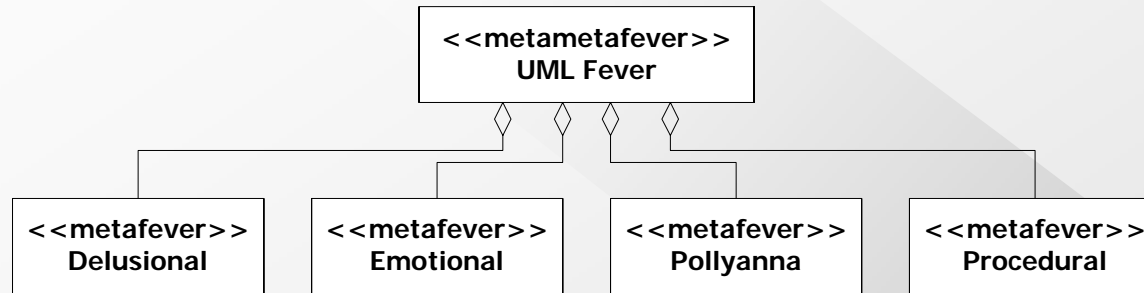
> **Death by UML Fever**

- Alex E. Bell, The Boeing Company
- ACM Queue vol. 2, no. 1 - March 2004
- Available on-line at www.acmqueue.org

> **Disclaimer**

... UML itself is not the direct cause of any maladies described herein. Instead, UML is largely an innocent victim caught in the midst of poor process, no process, or sheer incompetence of its users. Through no fault of its own, however, UML sometimes does amplify the symptoms of some fevers as the result of the often divine-like aura attached to it ...

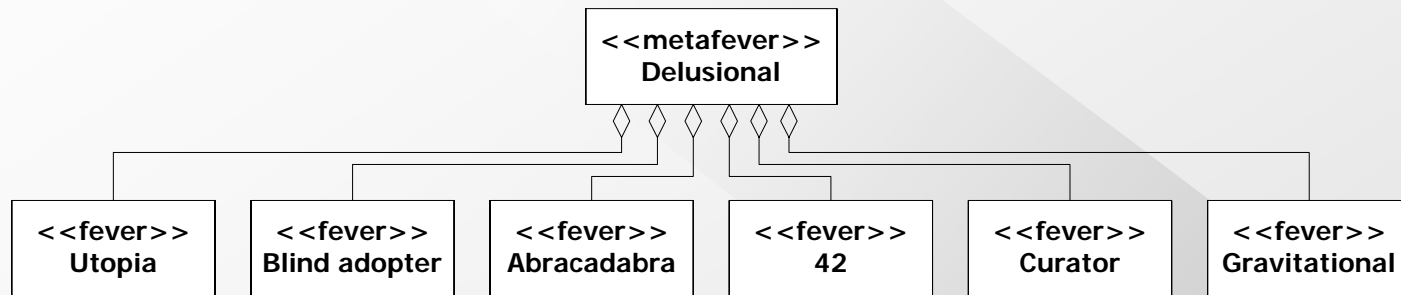
Meta-Fevers



> UML Fever aggregates:

- Delusional
- Emotional
- Pollyanna
- Procedural

Delusional Fevers



> Utopia

- Divine origins

> Blind adopter

- No tailoring required

> Abracadabra

- Full automation is here!

> 42

- UML is **The Answer**

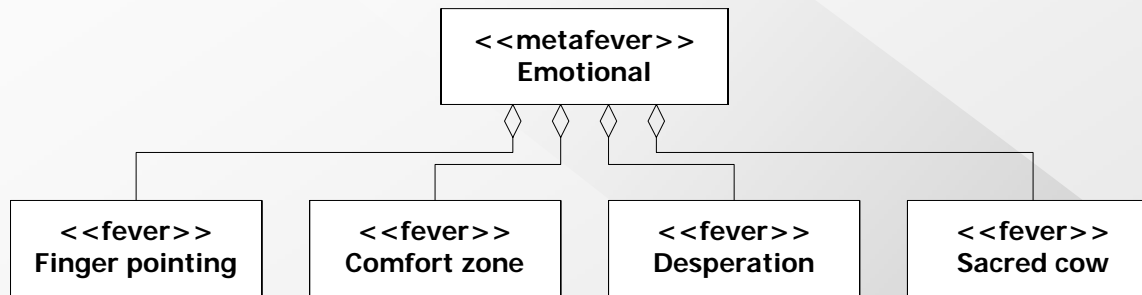
> Curator

- Absorption in diagrams

> Gravitational

- Mass of UML diagrams \neq value

Emotional Fevers



> Finger pointing

- Blame RUP or UML

> Comfort zone

- Tranquillity while modelling

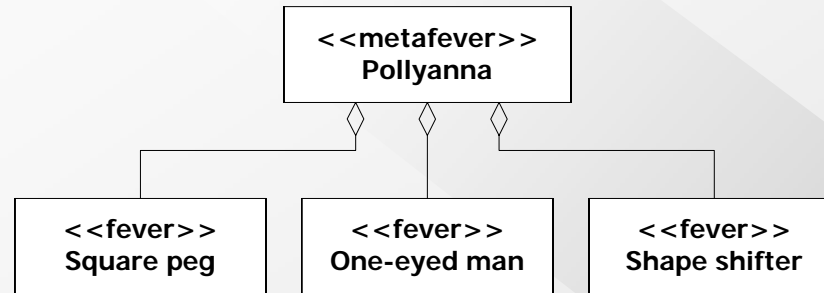
> Desperation

- The tool that cures all evils

> Sacred cow

- Let no diagram die

Pollyanna Fevers



> Square Peg

- Interchangeable staff

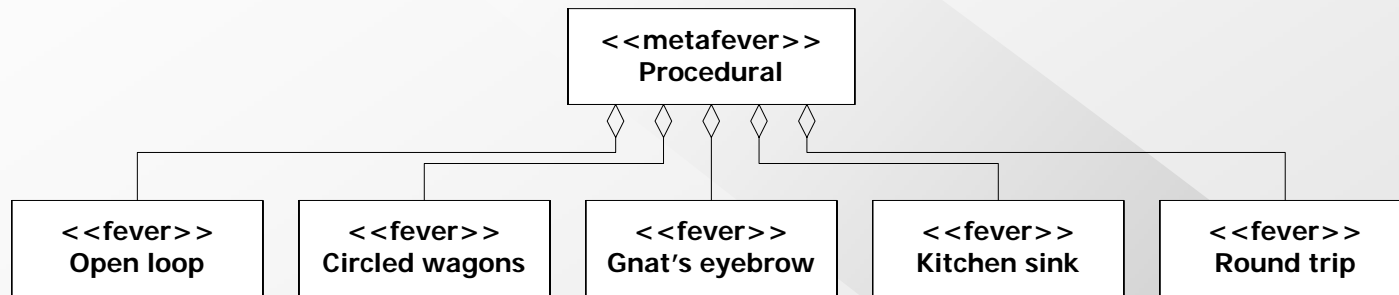
> One-eyed man

- In the realm of the blind

> Shape shifter

- One UML course does not a designer make

Procedural Fevers



> Open loop

- Just keep drawing

> Circled wagons

- Fine-grained functional decomposition of the domain

> Gnat's eyebrow

- Utmost detail

> Kitchen sink

- Model **Everything**, including...

> Round trip

- Abstraction deficiency
- Design \neq Implementation model

Conclusion

UNISYS

Imagine it • Done •

- > Systems Integration.
- > Outsourcing.
- > Infrastructure.
- > Server Technology.
- > Consulting.



Ceci n'est pas une pipe.

Questions



Thank you!