Data4MDE + FPVM Panel

availability (or rather, lack of it) of
large datasets and repositories of modelling artefacts

Panelists: Jordi Cabot, Davide di Ruscio,
Steven Kelly, Jesús Sánchez Cuadrado

22.6.2021

Steven Kelly

- Background:
  - Masters: Maths & Computer Science at Cambridge 1988–91
  - CASE tool SW dev. + year working as a field linguist in Kenya
  - PhD: Information Systems (MetaEdit+), Jyväskylä 1993–97
  - Multi-user, multi-paradigm, multi-tool, OO-repository-based
  - CTO at MetaCase, 1996–
- MetaCase founded from university research project 1991
  - Metamodeling & modelling tools: theory into practice
  - Domain-Specific Modeling: make new language for org/proj
- Fortunate to acquire large cases early on and build on that
  - 100s of metamodels, 100s GB of models, 1000s users
  - Plus those where info / models are not shared with us
- I love making tools and processes efficient and scalable

Questions & Opinions

- We all have so much to learn from each other!

1. In your opinion, why there are no large repositories/datasets of models?
   - The most valuable models are often the most secret. This won't change.
   - Those most interested in openness are often least interested in visual modelling
     - although there is at least one 'graphical' model by Linus!
2. What would be the socio-technical requirements for such repositories?
   - Public sector systems should be open source, including models
   - Need to solve the real-world problem of openness vs. vulnerability
   - Multi-user collaboration and versioning based on people, models; not text, code
3. How can we foster the modelling community to share their models?
   - Tooling to make it easy – and not just for tool developer ☺
   - Web platforms to offer a shared space and make it scale

Linus's state diagram in Linux kernel on GitHub

github.com/torvalds/linux/blob/master/net/ipv4/tcp_bbr.c#L17

```
 * Here is a state transition diagram for BBR:
 *
 *             |
 *             V
 *    +---> STARTUP  ----+
 *    |        |         |
 *    |        V         |
 *    |      DRAIN   ----+
 *    |        |         |
 *    |        V         |
 *    +---> PROBE_BW ----+
 *    |      ^    |      |
 *    |      |    |      |
 *    |      +----+      |
 *    |                  |
 *    +---- PROBE_RTT <--+
 *
 * A BBR flow starts in STARTUP, and ramps up its sending rate quickly.
 * When it estimates the pipe is full, it enters DRAIN to drain the queue.
 * In steady state a BBR flow only uses PROBE_BW and PROBE_RTT.
 * A long-lived BBR flow spends the vast majority of its time remaining
 * (repeatedly) in PROBE_BW, fully probing and utilizing the pipe's bandwidth
 * in a fair manner, with a small, bounded queue. *If* a flow has been
 * continuously sending for the entire min_rtt window, and hasn't seen an RTT
 * sample that matches or decreases its min_rtt estimate for 10 seconds, then
 * it briefly enters PROBE_RTT to cut inflight to a minimum value to re-probe
 * the path's two-way propagation delay (min_rtt). When exiting PROBE_RTT, if
 * we estimated that we reached the full bw of the pipe then we enter PROBE_BW;
 * otherwise we enter STARTUP to try to fill the pipe.
```