OSADL: Safety Critical Linux Working Group

Project # 8: Statistics - Model cleanup and analysis improvements

Rationale:

Uncovering systematic software development process faults by looking at metrics and attributes of the resulting code has been in use since the 1980s. The methods essentially consist of more or less well defined metrics, heuristics for attributes or indication of sound processes and trending for process stability. None of the metrics/attributes/trends are calibrated and it is questionable if they can be calibrated in any meaningful generic sense. However they possess the ability to indicate change. Furthermore trending and analysis of metrics development can pinpoint areas of intensified review and analysis. For the SIL2LinuxMP approach of greatest importance is the ability to detect process risks and/or code base of elevated risk that can be excluded from safety related systems and thus offers a high-level approach to risk elimination rather than focusing on technical measures for risk reduction only.

Statistical methods as introduced here do not try to "calculate" residual bugs or directly estimate risk - with other words this is not a proven-in-use (route 2_S) through the back-door - but rather we seek to minimize the risks by selection and - and this is crucial as it is a characteristics of complex systems - we need to extract an estimate for maintenance efforts so that monitoring and incident response planing is realistic. Complex safety related systems using pre-existing software elements must expect a significantly higher incident rate as well as change rate over time, compared to small traditional safety RTOS - statistical methods will be a key factor in planing realistically as well as evaluation of economic viability for a given system.

The methods proposed here are potentially of interest beyond safety related systems - both HA and business-critical systems may as well profit from such methods as could continuous improvement efforts for the kernel development life-cycle.

Overview of development:

- Phase 1:	Related data sources and tools • cregit - code evolution (token level authorship) • PASTA - email history on LKML, subsystem mailing lists • PIT/L2S - patch metrics and lines • DLCDM - commits, tools and developers
- Phase 2:	Overall process stability and evolution • Assessment of modification heuristics (Greg KH) • Simple correlation models (add/remove/mod) • Process evolution (tools and tags) • Commit mainlining process (tree hierarchy and review versions)

- *Phase 3:*	Bug survival time model(s) • Bug classification (heuristics) and survival-time of classes. • Automation and continuous update of prediction • Identification of distribution -> rationale
- *Phase 4:*	Developer "qualification" estimation • Maintainer model documentation • Contribution parameters (continuity, stability, focus, etc.) • Commits, reviews, testing, mailing lists (e.g. via PASTA)
- *Phase 5:*	Top-down model (commit meta-data) for -stable releases • Options beyond git-meta data (see phase 1) • Analysis of rc-3+ for inclusion • Zero truncated models using VGLM • Zero inflated models for subsystems and configs (investigation needed) • Merging of models from PIT data - Bayesian ? • Extend models with other data – Verification/plausibilitsation • Defined limiting criteria (parameter thresholds/dynamics)
- *Phase 6:*	Contingency planing • Residual but estimation • Uncertainty of prediction (sensitivity) • Subsystem/configuration specific sets classification (test-coverage e.g. inclusion in allnoconfig/defconfig(s) • Contingency weighting (metrics e.g. CC and history e.g. novelty/change-dynamics/developer-diversity/etc.)
- *Phase 7:*	Bottom-up model (configuration commit set summation) • Formalize model criteria (confidence, stability, sensitivity) • Non-parametric models and outliers handling • Causation: hypothetical causal networks • NNs (as second "channel") SOM....CNN
- *Phase 8:*	Review modeling methodology with academia • Define verification and validation criteria • ATC_modeling ? MODEL_AR: it is not clear if acceptance and test criteria might be reasonably generic, the current expectation is that assessment reports would be per model. • Critical review (e.g. Prof. Maurer ?) • Review with statistician TueV Rheinland • Submission to http://www.icse-conferences.org/

- *Phase 9:*	Training material and automation • Package models as R packages • Package reference data sets or (better) generator scripts based on publicly accessible meta-data (e.g. Linux-stable) • Basic modeling reference material (1 or 2 day course) • Specific model material for model types
- *Phase 10:*	Application to pre-defined kernel configuration in -lts or -cip • Effort recording (sufficient for planing) • Result recording (public access) • V&V efforts and recording of generated artifacts • Presentation/discussion to CA
- *Phase 11:*	Setup public accessible server and provide continuous prediction data for a small set of configuration and subset of architectures as well as subsystems • Development and user mailing lists
- *Phase 12:*	Formal acceptance for use in safety related systems • Formalized Acceptance and Test Criteria (ATC) • Assessment Report (AR) • Presentation at TueV Rheinland • Formal submission for acceptance as a basic safety mechanism in complex safety related systems. Formal proposal of statistical methods for use in Route 3_S
- *Phase 13:*	Dissemination and community integration • OSDAL/LF communication channels to industry • FLOSS/Linux community events and community channels (LWN) example commit classification and trigger events (historic events) • Safety community conferences
- *Possible extensions*	Additionally comparative work - pitching commercial Route1_S development against 3_S development would be of great interest. Not only would this allow a direct (quantitative or at least semi-quantitative comparison) but it would also serve as validation of the approach. Currently we do not see the willingness of commercial vendors in providing access to the necessary data though.
Participation form:	
	• Project development and data repository, R-package repository • Project mailing lists (development, review) • Basic training workshops • Dissemination workshops (may be in context of other events)
Responsibilities:	

	• Project legal entity: OSADL • Project resource maintenance OSADL Safety Critical Linux WG • Project Management: Nicholas Mc Guire • Technical lead: ....., Nicholas Mc Guire • Implementation: TBD, Nicholas Mc Guire • Certification Body: TueV Rheinland (proposed)
Effort estimation:	
	TODO
Timeline:	
Start	March 1, 2019
Milestone 1	Related tools and data preparation phase completed
Milestone 2	First simple model (e.g. bug-survival) prepared for discussion
Milestone 3	Top-down, qualification model prepared
Milestone 4	Bottom-up and qualification model presentation/discussion (this is a quite speculative model)
Milestone 5	Training material and R-package prepared
Milestone 6	CA Report and all related artifacts published
Maintenance phase	
Project format:	
	Partner funded project (OSADL members) - some of the models and methods may well be suitable beyond safety-related projects thus this project could be split into a non-safety and safety part.