Michael Lyu - Software Dependability Modeling with A Data-Driven AI Paradigm

Starts from a recap on the “traditional” software dependability modelling

- fault avoidance - removal - tolerance - prediction

Then question its relevance to model modern software systems, from service-oriented systems until cloud systems

- e.g., more subtle software failures, complex dynamic interactions, etc.

And it presents a paradigm shift towards “data”

> In the evolution of dependability modeling, the paradigm shift to a data-driven approach is an inevitable modeling effort, and AI techniques such as machine learning are called for.

Data-driven software dependability modeling moves:

- *from black box to white box*: white-box is preferable for modern software systems (distributed systems, clouds), because of their complex dependencies and failure patterns (which are also very unstable e.g., because of frequent software updates).

- *from model centric to data centric*: data centric models can mitigate incorrect assumptions on data distribution. We can use the output of monitoring of system logs and of log traces, topology at different layers, alerts

  → anomaly detection - failure detection - root cause analysis - failure prediction

- *from macro-level to micro-level*: the macro perspective is too coarse-grained (e.g., to capture dynamic interactions of microservices). Micro-level allows profiling the system and identifying anomalies.

- *from static analysis to dynamic analysis*: static analysis-based dependability modeling highly relies on historical failure data. But now we can use new data that we can collect and process: e.g., anomaly detector updates automatically with zero-shot/transfer learning.

Lionel Briand: Trustworthy Machine Learning-Enabled Systems

The talk reviews, from a personal perspective and experience, several challenges and techniques for automated testing of *software systems enabled by machine learning*, especially for their use in safety-critical systems

- **Testing is still the main mechanisms to which gain trusts**, but:
  - Various testing levels: test not just the DNN (as done by most research) but also the whole system, AND in the relevant scenario (e.g., as requested by SOTIF).
    - Scalability, realisms?
  - Huge input space (and labelling effort) and test suite adequacy
    - when is it good enough? (in dimension and sequences of inputs)
    - how can simulator be exploited/guided?
  - Information access: white-box and data-box are often not available. Test adequacy criteria require access to the DNN internals, often not applicable in practical settings
    - This way, we cannot rely on coverage criteria as neuron coverage

- Functional safety: is the uncertainty associated to the ML model acceptable?
  - Explaining test results (XAI)
- Failures
  - MLs failures result from both ML mispredictions and the effectiveness of countermeasures
- Robustness: adversarial attacks studied for robustness
  - but they are not realistic: adversarial “natural” inputs should be researched
- …
- Research directions:
  - Measure diversity of test inputs: the more diverse, the more likely to reveal faults (correlation between geometric diversity and faults is higher than surprise adequacy coverage and faults)
  - Cluster mispredictions to estimate faults on the DNN