Probabilistic Software Modeling

Hannes Thaller

Institute for Software Systems Engineering
Johannes Kepler University Linz

Summary

Probabilistic Software Modeling (PSM) [1] is a data-driven software modeling paradigm for analytical and generative methods in software engineering. PSM builds a Probabilistic Model (PM) from the system-under-inspection by analyzing its structure (i.e., types, properties, and executables), and behavior (i.e., runtime events), via static and dynamic code analysis.
A system's PM can be used, for example, to visualize the runtime behavior of code elements, generate tests, simulate the execution of subsystems, or to detect runtime anomalies.

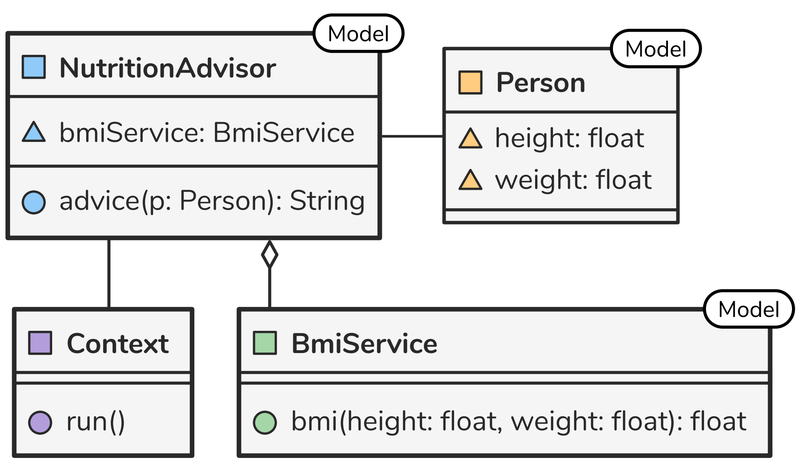
Applications

PSM builds statistical models of code elements from which new observations, based on the monitored runtime, can be sampled. This allows for predictive and generative application enabling in-depth **program comprehension**.

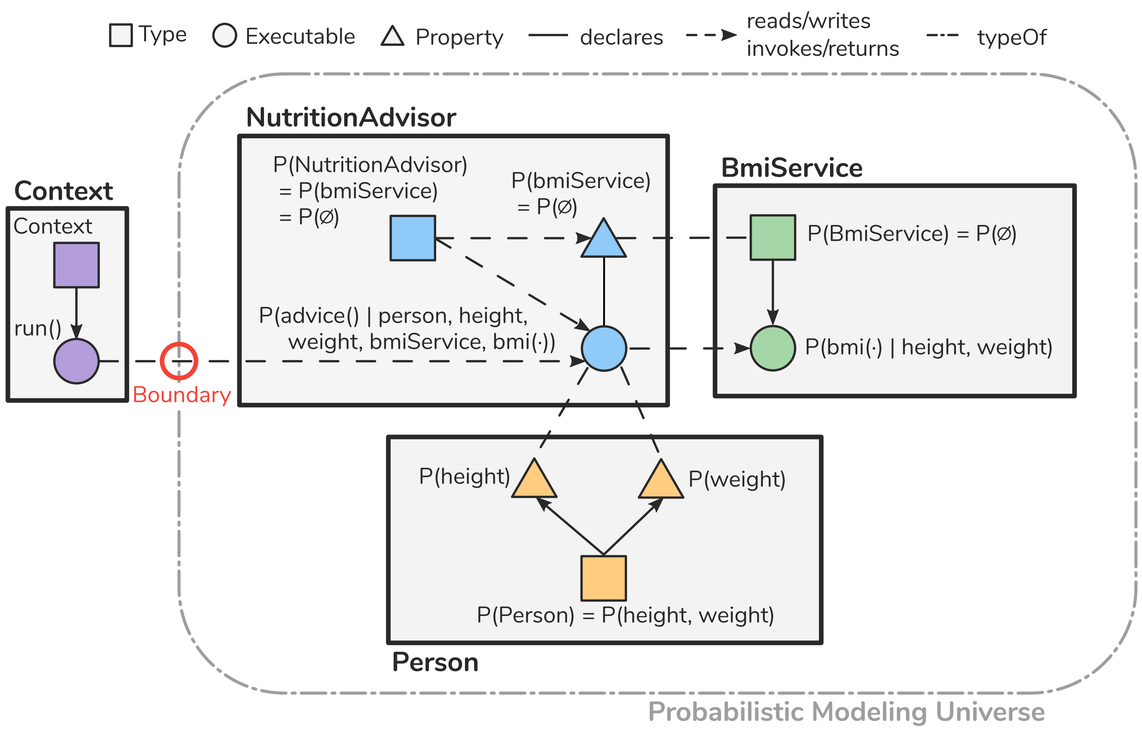
Predictive	**Example**
• Anomaly detection [2]	P(height, weight) < 0.10
• Comprehension & Visualization	P(weight | 160 < height < 161)
• Integrity and compatibility evaluation	$P(weight_{version1}) \sim P(weight_{version2})$
Generative	
• Test-case generation [3]	0.1 < P(bmi | height, weight)
• Simulation	new Person(height, weight) ~ P(weight, height)

Method

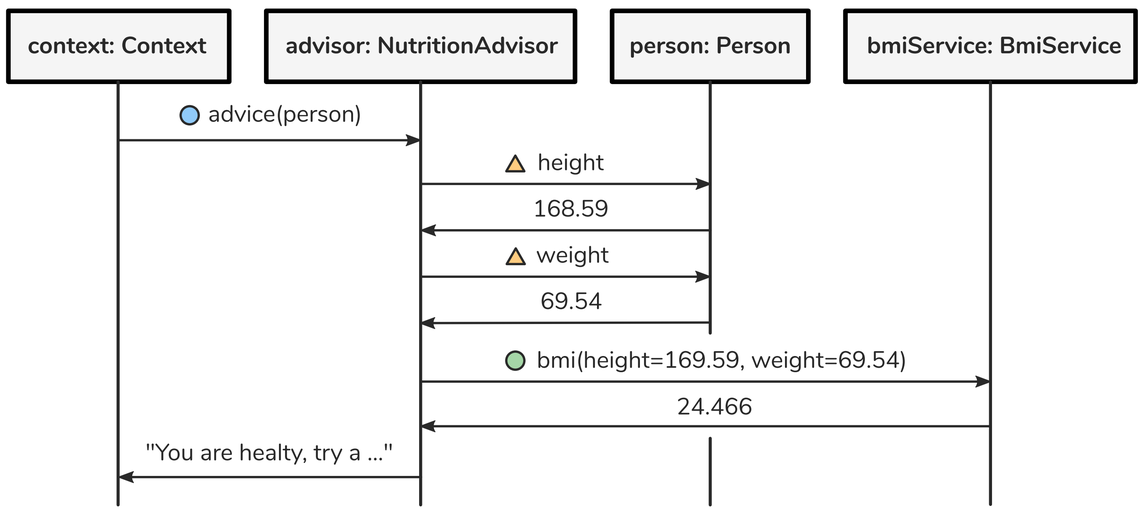
Software is inherently structured into concepts, states, and procedures. PSM analyzes this structure and uses it as a blueprint for the probabilistic model. In this case, the structural information comprises the classes *NutritionAdvisor*, *Person*, and *BmiService* along with their properties and executables.



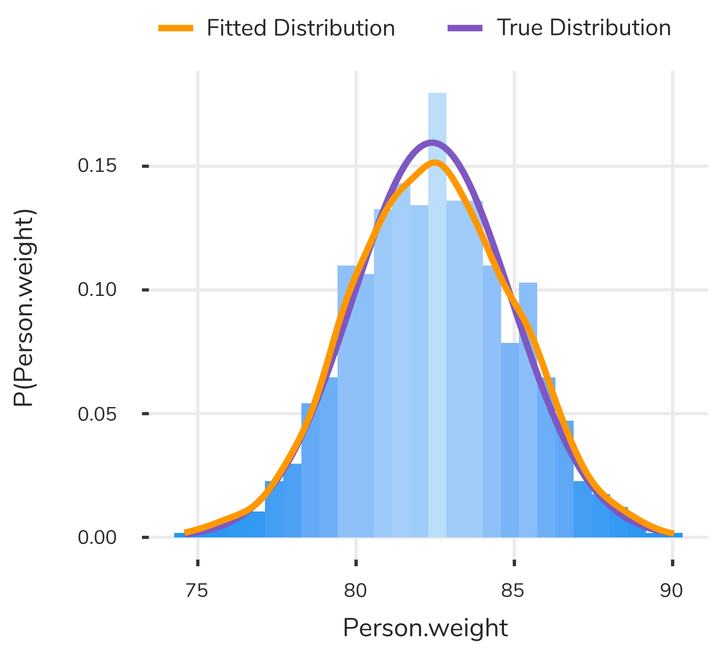
The program structure is transformed into a network of probabilistic models with similar interdependencies. Properties are variables. Executables are variables that are conditioned on their parameters, property reads and method invocations. Classes are the joint combination of their property variables.



The network of probabilistic models is fitted with the information collected from the runtime monitoring of the program providing its behavior.



Finally, the models may generate new observations based on the Fitted Distribution fuelling additional inference and analysis tasks. For example, visualization enables program behavior comprehension on a new level.



References

[1] H. Thaller, “Probabilistic Software Modeling,” arXiv:1806.08942 [cs], Jun. 2018.
[2] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, Jul. 2009.
[3] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, “A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation,” IEEE Transactions on Software Engineering, vol. 36, no. 6, pp. 742–762, Nov. 2010.

The research reported in this poster has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

Contact

Hannes Thaller
Institute for Software Systems Engineering
Johannes Kepler University Linz
Altenberger Straße 69
+43 732 2468 4395
hannes-thaller.com
hannes.thaller@jku.at